THE EQUATIONAL THEORIES PROJECT: USING LEAN AND GITHUB TO COMPLETE AN IMPLICATION GRAPH IN UNIVERSAL ALGEBRA

MATTHEW BOLAN, JOSE BROX, MARIO CARNEIRO, MARTIN DVOŘÁK, ANDRÉS GOENS, HARALD HUSUM, ZOLTAN KOCSIS, ALEX MEIBURG, PIETRO MONTICONE, DAVID RENSHAW, JÉRÉMY SCANVIC, SHREYAS SRINIVAS, TERENCE TAO, ANAND RAO TADIPATRI, VLAD TSYRKLEVICH, DANIEL WEBER, FAN ZHENG, ... (IN ALPHABETICAL ORDER)

Abstract. We report on the *Equational Theories Project* (ETP), an online collaborative pilot project to explore new ways to collaborate in mathematics with machine assistance. The project sought to determine the implication graph between 4694 equational laws on magmas, by a combination of human-generated and automated proofs, all validated by the formal proof assistant language *Lean*. **state key outcomes**

Contents

Date: December 16, 2024.

1. Introduction

The purpose of this paper is to report on the *Equational Theories Project* (ETP)[1], a pilot project launched[2] in September 2024 to explore new ways to collaboratively work on mathematical research projects using machine assistance. The project goal, in the area of universal algebra, was selected[3] to be particularly amenable to crowdsourced and computer-assisted techniques, while still being of mathematical research interest. **Describe outcomes**

1.1. **Magmas and Equational Laws.** In order to describe the mathematical goals of the ETP, we need some notation. A *magma* $\mathcal{M} = (M, \diamond)$ is a set M (known as the *carrier*) together with a binary operation $\diamond \colon M \times M \to M$. An *equational law* for a magma, or *law* for short, is an identity involving $\diamond$ and some formal indeterminates, which we will typically denote using the Roman letters $\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{w}, \mathrm{u}, \mathrm{v}$, as well as the formal equality symbol $\simeq$ in place of the equality symbol $=$ to emphasize the formal nature of the law.

In the ETP, a unique number was assigned to each equational law, via a numbering system that we describe in Appendix A. For instance, the *commutative law* $\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{y} \diamond \mathrm{x}$ is assigned the equation number (E43), while the *associative law* $(\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{z} \simeq \mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z})$ is assigned the equation number (E4512). A list of all equations referred to by number in this paper is provided in Appendix A.

A magma $\mathcal{M} = (M, \diamond)$ obeys a law E if the law E holds for all possible assignments of the indeterminate to elements of M, in which case we write $\mathcal{M} \models E$. Thus, for instance $\mathcal{M} \models E43$ if one has $x \diamond y = y \diamond x$ for all $x, y \in M$. Note that the formal indeterminate symbols x, y in $E43$ are now replaced by concrete elements x, y of the carrier M.

We say that a law E *entails* or *implies* another law E' if every magma that obeys E, also implies E': $(\mathcal{M} \models E) \implies (\mathcal{M} \models E')$. We write this relation as $E \vdash E'$. We say that two laws are *equivalent* if they entail each other. For instance, the constant law $\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{z} \diamond \mathrm{w}$ (E46) can easily be seen to be equivalent to the law $\mathrm{x} \diamond \mathrm{x} \simeq \mathrm{y} \diamond \mathrm{z}$ (E41). It is easy to see that $\vdash$ is a pre-order, that is to say a partial order after one quotients by equivalence.

In this entailment pre-ordering, the maximal element is given by the trivial law $\mathrm{x} \simeq \mathrm{x}$ (E1), and the minimal element is given by the singleton law $\mathrm{x} \simeq \mathrm{y}$ (E2), thus $E2 \vdash E \vdash E1$ for all laws E.

[1] https://teorth.github.io/equational_theories/
[2] https://terrytao.wordpress.com/2024/09/25
[3] The specific mathematical goal was inspired by a MathOverflow question.

We also define a variant: we say that E *entails* E' *for finite magmas*, and write $E \vdash_{\text{fin}} E'$, if every *finite* magma M that obeys E, also obeys E'. Clearly, the relation $E \vdash E'$ implies $E \vdash_{\text{fin}} E'$; but, as observed by Austin [2], the converse is not true in general.

The *order* of an equational law is the number of occurrences of the magma operation. For instance, the commutative law (E43) has order 2, while the associative law (E4512) has order 4. We note some selected laws of small order that have previously appeared in the literature:

- The *central groupoid law* $\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond \mathrm{z})$ (E168) is an order-3 law introduced by Evans [10] and studied further by Knuth [18] and many further authors, being closely related to central digraphs (also known as unique path property diagraphs), and leading in particular to the discovery of the Knuth-Bendix algorithm [19]; see [24] for a more recent survey.
- *Tarski's axiom* $\mathrm{x} \simeq \mathrm{y} \diamond ((\mathrm{z} \diamond (\mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z}))))$ (E543) is an order-4 law that was shown by Tarski [37] to characterize the operation of subtraction in an abelian group; that is to say, a magma $\mathcal{M} = (M, \diamond)$ obeys (E543) if and only if there is an abelian group structure on $\mathcal{M}$ for which $x \diamond y = x - y$ for all $x, y \in M$.
- In a similar vein, it was shown in [32] (see also [33]) that the order-4 law $\mathrm{x} \simeq (\mathrm{y}\diamond\mathrm{z})\diamond(\mathrm{y}\diamond(\mathrm{x}\diamond\mathrm{z}))$ (E1571) characterizes addition (or subtraction) in an abelian group of exponent 2; it was shown in [30] that the order-6 law $\mathrm{x} \simeq (\mathrm{y}\diamond((\mathrm{x}\diamond\mathrm{y})\diamond\mathrm{y}))\diamond(\mathrm{x}\diamond(\mathrm{z}\diamond\mathrm{y}))$ (E345169) characterizes the Sheffer stroke in a boolean algebra, and it was shown in [13] that the order-8 law $\mathrm{x} \simeq \mathrm{y} \diamond ((((\mathrm{y} \diamond \mathrm{y}) \diamond \mathrm{x}) \diamond \mathrm{z}) \diamond (((\mathrm{y} \diamond \mathrm{y}) \diamond \mathrm{y}) \diamond \mathrm{z}))$ (E42323216) characterizes division in a (not necessarily abelian) group.

Some further examples of laws characterizing well-known algebraic structures are listed in [29].

The Birkhoff completeness theorem [3, Th. 3.5.14] implies that an implication $E \vdash E'$ of equational laws holds if and only if the left-hand side of E' can be transformed into the right-hand side by a finite number of substitution rewrites using the law E. However, the problem of determining whether such an implication holds is undecidable in general [31]. Even when the order is small, some implications[4] can require lengthy computer-assisted proofs; for instance, it was noted in [15] that the order-4 law $\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond ((\mathrm{x} \diamond \mathrm{z}) \diamond \mathrm{z})$ (E1689) was equivalent to the singleton law (E2), but all known proofs are computer-assisted. Furthermore, for the finite magma implication relation $E \vdash_{\text{fin}} E'$, no analogue of the Birkhoff completeness theorem is available.

1.2. **The Equational Theories Project.**

As noted in Appendix A, there are 4694 equational laws of order at most 4. The primary mathematical goal of the ETP was to completely determine the *implication graph* for these laws, in which there is a directed edge from E to E' precisely when $E \vdash E'$. As the project progressed, an additional goal was added to determine the slightly larger *finite implication graph*, in which there is a directed edge from E to E' precisely when $E \vdash_{\text{fin}} E'$.

[4] Another contemporaneous example of this phenomenon was the solution of the Robbins problem [28].

Such systematic determinations of implication graphs have been seen previously in the literature; for instance, in [34], the relations between 60 identities of Bol–Moufang type were established, and in the blog post [40, §17], some initial steps towards generating this graph for the first hundred or so laws on our list were performed. However, to our knowledge, the ETP is the first project to study such implications at the scale of thousands of laws.

The ETP requires the determination of the truth or falsity of $4694^2 = 22033636$ implications (for both arbitrary magmas and finite magmas); while one can use properties such as the transitivity of entailment to reduce the work somewhat, this is clearly a task that requires significant automation. It was also a project highly amenable to crowdsourcing, in which different participants could work on developing different techniques, each of which could be used to fill out a different part of the implication graph. In this respect, the project could be compared with a Polymath project [12], which used online forums such as blogs and wikis to openly collaborate on a mathematical research problem. However, the Polymath model required human moderators to review and integrate the contributions of the participants, which clearly would not scale to the ETP which required the verification of over twenty million mathematical statements. Instead, the ETP was centered around a GitHub repository in which the formal mathematical contributions had to be entered in the proof assistant language *Lean*, where they could be automatically verified. In this respect, the ETP was more similar to the recently concluded Busy Beaver Challenge[5], which was a similarly crowdsourced project that computed the fifth Busy Beaver number $BB(5)$ to be 47176870 through an analysis of about 180 million Turing machines, with the halting analysis being verified in a variety of computer languages, with the final formal proof written in the proof assistant language *Coq*. One of the aims of the ETP was to explore potential workflows for such collaborative, formally verified mathematical research projects that could serve as a model for future projects of this nature.

Secondary aims of the ETP included the possibility of discovering unusually interesting equational laws, or new experimental observations about such laws, that had not previously been noticed in the literature; and to develop benchmarks to assess the performance of automated theorem provers and other AI tools.

1.3. **Outcomes. This text assumes (optimistically) that both the original and finite implication graph will be completely formalized.**

The ETP achieved its primary objectives, with all of the implications for both arbitrary magmas and finite magmas formalized in the proof assistant language *Lean*, and can be found on the ETP GitHub repository. See Figure 1, Figure 2 for some small fragments of the implication graphs produced. The experience of running such a large collaborative research project introduced several challenges, which we report upon in Section 4. Also, a variety of methods with varying degrees of automation or computer-assistance had to be developed to resolve all the implications, which had quite a variety of difficulty levels.

[5] `https://bbchallenge.org/`

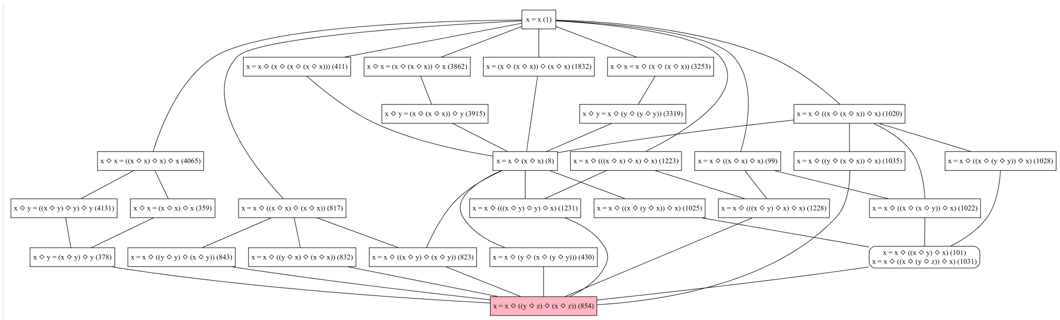

FIGURE 1. A Hasse diagram of all the equational laws implied by (E854) (for unrestricted magmas). An edge in this diagram indicates that the lower equation implies the higher one. Rounded rectangles indicate groups of equivalent laws. This graph was produced by the visualization tool *Graphiti*, which was developed for this project.

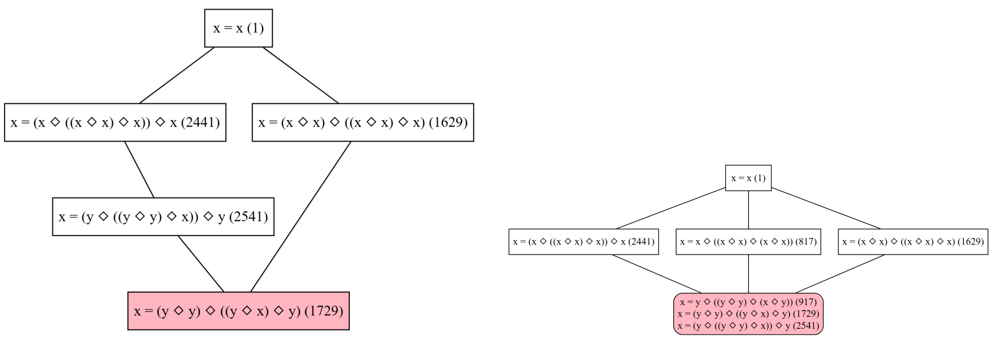

FIGURE 2. A Hasse diagram of all the equational laws implied by (E1729), both for unrestricted magmas (left) and finite magmas (right). Note the slightly larger number of implications in the latter.

Of the 22033636 possible implications $E \vdash E'$, 8178279 (or 37.12%) would end up being true; for a slightly larger set **give statistics**, the weaker implication $E \vdash_{\text{fin}} E'$ held. To establish such positive implications $E \vdash E'$ or $E \vdash_{\text{fin}} E'$, the main techniques used were as follows:

- A very small number of positive implications were established and formalized by hand, mostly through direct rewriting of the laws; but this approach would not scale to the full project.
- Simple rewriting rules, for instance based on the observation that any law of the form $\mathrm{x} \simeq f(\mathrm{y}, \mathrm{z}, \dots)$ was necessarily equivalent to the trivial law (E2), could already reduce the size of potential equivalence classes by a significant fraction. We discuss this method in Section 6.1.
- The preorder axioms for $\vdash$, as well as the "duality" symmetry of the preorder with respect to replacing a magma operation $x \diamond y$ with its opposite $x \diamond^{\text{op}} y := y \diamond x$, can be used to significantly cut down on the number of implications that need to be proven explicitly; ultimately, only 10657 (0.05%) of the positive implications needed a direct proof. **Update these stats when we obtain our final theorem**

- To obtain additional implications for finite magmas, heavy reliance was made on the fact that for functions $f\colon M \to M$ on a finite set M, surjectivity was equivalent to injectivity. Some more sophisticated variants of this idea can lead to additional implications; see Section 5.1.
- Automated Theorem Provers (ATP) could be deployed at extremely fast speeds to establish a complete generating set of positive implications; see Section 7.

More challenging were the 13855357 (62.88%) implications that were false, $E \not\vdash E'$, and particularly the slightly smaller set of **give stats here** implications that were false even for finite magmas, $E \not\vdash_{\mathrm{fin}} E'$. Here, the range of techniques needed to refute such implications were quite varied.

- Syntactic methods, such as observing a "matching invariant" of the law E that was not shared by the law E', could be used to obtain some refutations. For instance, if both sides of E had the same order, but both sides of E' did not, this could be used to syntactically refute $E \vdash E'$. Similarly, if the law E was confluent, enjoyed a complete rewriting system, or otherwise permitted some understanding of the free magma associated to that law, one could decide the assertions $E \vdash E'$ for all possible laws E', or at least a significant fraction of such laws. We discuss these methods, and the extent to which they can be automated in Section 6.
- Small finite magmas, which can be described explicitly by multiplication tables, could be tested by brute force computations to provide a large number of finite counterexamples to implications, or by ATP-assisted methods. See Section 5.1.
- Linear models, in which the magma operation took the form $x \diamond y = ax + by$ for some (commuting or non-commuting) coefficients a, b, allowed for another large class of counterexamples to implications, which could be automatically scanned for either by brute force or by Grobner basis type calculations; many of these examples could also be made finite. See Section 5.2.
- Translation invariant models, in which the magma operation took the form $x \diamond y = x + f(y - x)$ on an additive group, or $x \diamond y = xf(x^{-1}y)$ on a non-commutative group, reduce matters to analyzing certain functional equations; see Section 5.3.
- Greedy methods, in which either the multiplication table $(x, y) \mapsto x \diamond y$ or the function f determining a translation-invariant model are iteratively constructed by a greedy algorithm subject to a well-chosen ruleset, were effective in resolving many implications not easily disposed of by preceding methods. See Section 5.5.
- Starting with a simple base magma $\mathcal{M}$ obeying both E and E', and either enlarging it to a larger magma $\mathcal{M}'$ containing $\mathcal{M}$ as a submagma, extending it to a magma $\mathcal{N}$ with a projection homomorphism $\pi : \mathcal{N} \to \mathcal{M}$, or modifying the multiplication table on a small number of values, also proved effective when combined with greedy methods or with a "magma cohomology" construction. See Section 5.6.
- To each equation E one can associate a "twisting semigroup" S_E. If S_E is larger than $S_{E'}$, then this can often be used to disprove the implication $E \vdash E'$; see Section 5.4.
- Some *ad hoc* models based on existing mathematical objects, such as infinite trees, rings of polynomials, or "Kisielewicz models" utilizing the prime factorization of the natural numbers, could also handle some otherwise difficult cases. In some cases, the magma law induced some relevant and familiar structures, such as a directed graph

or a partial order, which also helped guide counterexample constructions. We will not detail these diverse examples here, but refer the reader to the ETP blueprint for more discussion.
- Automated theorem provers were helpful in identifying which simplifying axioms could be added to the magma without jeopardizing the ability to refute the desired implication $E \vdash E'$ or $E \vdash_{\mathrm{fin}} E'$.

In the course of completing the implication graph, some interesting new algebraic structures were discovered. One such example concerns the magmas obeying (E1485), which we refer to as *weak central groupoids* as they contain the central groupoids (obeying (E168)) as a subclass. In [18] it was observed that all finite central groupoids have order equal to a perfect square n^2; empirically, we have found that finite weak central groupoids always have order n^2 or $2n^2$, although we have no rigorous proof of this claim; they also have a graph-theoretic interpretation analogous to the interpretation of central groupoids as digraphs with the unique path property. For these and other observations we refer the reader to the blueprint of the ETP.

The objective of using the data from the ETP to establish well-calibrated benchmarks to evaluate ATPs remains an interesting open problem; the participants of this project did not have the required expertise to develop and test such benchmarks to the standards expected in the area. However, in Section 7 we present a more informal "field report" of our experiences using ATPs in the project, in the hope that this will provide some useful guidance to other researchers seeking to incorporate ATPs into their own research.

1.4. **Further directions.**

While the primary objective of the ETP was being completed, some additional related results were generated as spinoffs. Specifically:

- In the blueprint on the ETP web site, we report some partial progress on we report on classifying which of the 57882 distinct laws of order 5 are equivalent to the singleton law (E2), either with or without the requirement that the magma be finite.
- In Section 9 we report on classifying the laws of order 8 that are equivalent to the Higman-Neumann law (E42323216).

Also mention ML stuff, GUI

2. Notation and Mathematical Foundations

If $\mathcal{M} = (M, \diamond)$ is a magma, we define the left and right multiplication operators $L_a, R_a \colon M \to M$ for $a \in M$ by the formula

$$L_x y = R_y x := x \diamond y. \tag{1}$$

We also define the squaring operator $S \colon M \to M$ by

$$Sx := x \diamond x = L_x x = R_x x. \tag{2}$$

A *homomorphism* $f\colon \mathcal{M} \to \mathcal{M}'$ between two magmas $\mathcal{M} = (M, \diamond)$, $\mathcal{M}' = (M', \diamond')$ is a function $f\colon M \to M'$ such that $f(x \diamond y) = f(x) \diamond' f(y)$ for all $x, y \in M$. An *isomorphism* is a homomorphism that is invertible (which implies that the inverse is also a homomorphism). An *endomorphism* is a homomorphism from a magma to itself.

If X is an alphabet, we let $\mathcal{M}_X$ denote the free magma generated by X, thus an element of $\mathcal{M}_X$ is either a letter in X, or of the form $w_1 \diamond w_2$ with $w_1, w_2 \in \mathcal{M}_X$. Every function $f\colon X \to M$ into a magma $\mathcal{M} = (M, \diamond)$ extends to a unique homomorphism $\varphi_f\colon \mathcal{M}_X \to \mathcal{M}$. Formally, an equational law with some indeterminates in X can be written as $w_1 \simeq w_2$ for some $w_1, w_2 \in X$; a magma $\mathcal{M} = (M, \diamond)$ then obeys this law if and only if $\varphi_f(w_1) = \varphi_f(w_2)$ for all $f\colon X \to M$. We also define the order of a word $w \in \mathcal{M}_X$ to be the number of occurrences of $\diamond$ in the word, thus letters in X are of order 0, and the order of $w_1 \diamond w_2$ is the sum of the orders of w_1, w_2.

A *theory* is a collection Γ of equational laws; we say that a magma $\mathcal{M}$ *satisfies* a theory, and write $\mathcal{M} \models \Gamma$, if every law in Γ is obeyed by $\mathcal{M}$. If E is an equational law, we write $\Gamma \vdash E$ if every magma that satisfies Γ also satisfies E. A *free magma* $\mathcal{M}_{X,\Gamma}$ for such a theory Γ and an alphabet X is a magma satisfying Γ together with a map $\iota_{X,\Gamma}\colon X \to \mathcal{M}_{X,\Gamma}$ which is universal in the sense that every function $f\colon X \to \mathcal{M}$ to a magma $\mathcal{M}$ satisfying Γ uniquely determines a homomorphism $\varphi_{f,\Gamma}\colon \mathcal{M}_{X,\Gamma} \to \mathcal{M}$ such that $\phi_{f,\Gamma} \circ \iota_{X,\Gamma} = f$. This magma is unique up to isomorphism; a canonical way to construct it is as the quotient $\mathcal{M}_X/\sim_\Gamma$ of the free magma $\mathcal{M}_X$ by the equivalence relation $\sim_\Gamma$ given by declaring $w \sim_\Gamma w'$ if $\Gamma \vdash w \simeq w'$. If $\Gamma = \{E\}$ consists of a single law E, we write $\mathcal{M}_{X,E}$, $\sim_E$, $\varphi_{f,E}$ for $\mathcal{M}_{X,\{E\}}$, $\sim_{\{E\}}$, $\varphi_{f,\{E\}}$ respectively . **Give reference for free magmas relative to theories**

In general, the free magma $\mathcal{M}_{X,\Gamma}$ is difficult to describe in a tractable form, but for some theories, one has a simple description:

Example 2.1 (Commutative and associative free magma)**.** The free magma $\mathcal{M}_{X,\{E43,E4512\}}$ for the commutative law (E43) and the associative law (E4512) is the free abelian semigroup generated by X (with $\iota_{X,\{E43,E4512\}}$ the obvious embedding map).

Example 2.2 (Left-absorptive free magma)**.** The free magma $\mathcal{M}_{X,\{E4\}}$ for the left-absorptive law (E4) is the magma with carrier X and operation $x \diamond y = x$ (with ι_{X,E_4} the identity).

Every magma $\mathcal{M}$ has an opposite $\mathcal{M}^{\mathrm{op}}$, which has the same carrier but the opposite operation $x \diamond^{\mathrm{op}} y := y \diamond x$. A magma $\mathcal{M}$ obeys an equational law E if and only if its opposite $\mathcal{M}^{\mathrm{op}}$ obeys the dual law E^*, defined by reversing the all operations. For instance, the dual of $\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z})$ (E327) is $\mathrm{y} \diamond \mathrm{x} \simeq (\mathrm{z} \diamond \mathrm{y}) \diamond \mathrm{x}$, which in our numbering system we rewrite in normal form as $\mathrm{x} \diamond \mathrm{y} \simeq (\mathrm{z} \diamond \mathrm{x}) \diamond \mathrm{y}$ (E395).

We then see that the implication graph has a duality symmetry: given two equational laws E_1, E_2, we have $E_1 \vdash E_2$ if and only if $E_1^* \vdash E_2^*$.

3. Formal Foundations

TODO: expand this sketch.

Here we describe the Lean framework used to formalize the project, covering technical issues such as:

- Magma operation symbol issues
- Syntax (‘LawX‘) versus semantics (‘EquationX‘)
- ”Universe hell” issues
- Additional verification (axiom checking, Leanchecker, etc.)
- Use of the ‘conjecture‘ keyword
- Use of namespaces to avoid collisions between contributions. (Note: we messed up with this with FreeMagma! Should have namespaced back end results as well as front end ones.)
- Use of Facts command to efficiently handle large numbers of anti-implications at once

Upstream contributions:

- Mathlib contributions
- LeanBlueprint contributions

4. Project Management

TODO: expand this sketch.

Shreyas Srinivas and Pietro Monticone have volunteered to take the lead on this section.

Discuss topics such as:

- Project generation from template
- GitHub issue management with labels and task management dashboard
- Continuous integration (builds, blueprint compilation, task status transition)
- Pre-push git hooks
- Use of blueprint (small note, see #406: blueprint chapters should be given names for stable URLs)
- Use of Lean Zulip (e.g. use of polls)

Maybe give some usage statistics, e.g. drawing from `https://github.com/teorth/equational_theories/actions/metrics/usage`

Mention that FLT is also using a similar workflow.

4.1. Handling Scaling Issues.

Also mention some early human-managed efforts (”outstanding tasks”, manually generated Hasse diagram, etc.) which suffices for the first one or two days of the project but rapidly became unable to handle the scale of the project.

Mention that some forethought in setting up a GitHub organizational structure with explicit admin roles etc. may have had some advantages if we had done so in the planning stages of the project, but it was workable without this structure (the main issue is that a single person – Terry – had to be the one to perform various technical admin actions).

Use of transitive reduction etc. to keep the Lean codebase manageable. Note that the project is large enough that one cannot simply accept arbitrary amounts of Lean code into the codebase, as this could make compilation times explode. Also note somewhere that transitive completion can be viewed as directed graph completion on a doubled graph consisting of laws and their formal negations.

Technical debt issues, e.g., complications stemming from an early decision to make Equations.lean and AllEquations.lean the ground truth of equations for other analysis and visualization tools, leading to the need to refactor when AllEquations.lean had to be split up for performance reasons.

Note that the "blueprint" that is now standard for guiding proof formalization projects is a bit too slow to keep up with this sort of project that is oriented instead about proving new results. Often new results are stated and formalized without passing through the blueprint, which is then either updated after the fact, or not at all. So the blueprint is more of an optional auxiliary guiding tool than an essential component of the workflow.

4.2. **Other Design Considerations.** Explain what "trusting" Lean really means in a large project. Highlight the kind of human issues that can interfere with this and how use of tools like external checkers and PR reviews by people maintaining the projects still matters. Provide guidelines on good practices (such as branch protection or watching out for spurious modifications in PRs, for example to the CI). Highlight the importance of following a proper process for discussing and accepting new tasks, avoiding overlaps etc. These issues are less likely to arise in projects with one clearly defined decision maker as in this case, and more likely to arise when the decision making has to be delegated to many maintainers.

Note that despite the guarantees provided by Lean, non-Lean components still contained bugs. For instance, an off-by-one error in an ATP run created a large number of spurious conjectures, and some early implementations of duality reductions (external to Lean) were similarly buggy. "Unit tests", e.g., checking conjectured outputs against Lean-validated outputs, or by theoretical results such as duality symmetry, were helpful, and the Equation Explorer visualization tool also helped human collaborators detect bugs.

Meta: documenting thoughts for the future record is quite valuable. It would be extremely tedious to try to reconstruct real-time impressions long after the fact just from the GitHub commit history and Zulip chat archive.

4.3. **Maintenance.** Describe the role of maintainers and explain why they need to be conversant in the mathematics being formalised, as well as Lean. As such, the role of maintainers is often akin to postdocs or assistant profs in a research group who do some research of their own, but spend much of their time to guide others in performing their tasks, the key difference being that contributors are volunteers who choose their own tasks. Explain the tasks maintainers must perform. Examples:

- Reviewing proofs,
- Helping with proofs and theorem statements when people get stuck,
- Offering suggestions and guidance on how to produce shorter or more elegant proofs,
- Ensuring some basic standards are met in proof blocks that make proofs robust to upstream changes,
- Creating and maintaining CI processes,
- Responding to task requests,
- Evaluating theorem and definition formulations (for example unifying many theorem statements into one using FactsSyntax),
- Suggesting better ones where possible,
- Ensuring that there is no excessive and pointless overlap of content in different contributions (TODO: elaborate on what level of overlap was permissible and what we consider excessive).

5. Counterexample constructions

In this section we collect the various techniques developed in the ETP to construct counterexamples to various implications $E \vdash E'$.

5.1. **Finite magmas.**

A finite magma $\mathcal{M}$ of order n can be labeled by the carrier $\{1, \dots, n\}$ and described by specifying the multiplication table $\diamond\colon \{1, \dots, n\} \times \{1, \dots, n\} \to \{1, \dots, n\}$. By generating a list of all the equational laws En, $n = 1, \dots, 4694$ obeyed by this magma, one can create refutations: if $\mathcal{M} \models En$ and $\mathcal{M} \not\models Em$, then clearly $E_n \not\vdash_{\mathrm{fin}} E_m$ and hence also $E_n \not\vdash E_m$. It is feasible to brute force over all $\sum_{n=2}^{4} n^{n^2} \approx 4.3 \times 10^9$ non-trivial magmas of order at most 4 to obtain many refutations of this type. By performing brute force over all magmas up to order 4, a total of $13,632,566$ implications (61.9% of all implications, and 96.3% of the false ones) can be refuted with 524 distinct magmas. Of these implications, $13,345,053$ were refuted with 3×3 magmas, with the remaining $415,293$ requiring 4×4 magmas. Performing this search took 165 CPU-hours.

However, it is not feasible to exhaustively search over the $5^{5^2} \approx 3 \times 10^{17}$ magmas of order 5, even after quotienting out by isomorphism and symmetry (which roughly saves a factor of $5! \times 2 = 240$). Randomly sampling such magmas did not produce significant refutations, as random magmas of order 5 tended to obey few laws, and the set of laws covered were usually also exhibited by smaller magmas. A more fruitful approach was to randomly sample from magmas with additional properties that encouraged satisfiability of a greater set of laws. These included linear and quadratic magmas (discussed below), and cancellative magmas. On the other hand, some classes of magmas, such as commutative magmas, ended up producing a disappointingly small number of additional refutations.

For specific refutations, it was sometimes possible to locate a finite example with an ATP, particularly if one also imposed additional axioms (e.g., an idempotence axiom $x = x \diamond x$) that one suspected would be useful. For medium-sized magmas (of order $n = 5, 6, 7, 8$), this appeared to be a more efficient approach than brute force exhaustion of all such magmas.

Order of smallest non-trivial model	Number of laws
Trivial only	1496
2	3136
3	32
4	14
5	14
7	2

TABLE 1. Number of laws of order at most 4 whose smallest non-trivial model is of a given size.

Discuss this further, perhaps give an example, or refer to the ATP section. See also the discussion threads "Counterexamples by enumerating words in quotient magmas" and "Using SAT solvers for model generation" threads (sorry, LaTeX is choking on the URLs for some reason).

It is a result of Kisielewicz [16] that every law En with $n \leq 4694$ is either equivalent to the singleton law $E2$, or else has a non-trivial finite model; in other words, the implications $En \vdash E2$ and $En \vdash_{\mathrm{fin}} E2$ are equivalent for $n \leq 4694$. In fact our brute force search revealed that in the latter case there is always a model of order $2 \leq n \leq 5$, with the lone exception of (E1286) (and its dual (E2301)), for which the smallest non-trivial finite model was of order 7, as presented in Example 5.1 below. In fact, most of the 4694 laws either only had trivial models, or had an order 2 model, as shown in Table 1.

5.2. **Linear models.** A fruitful source of counterexamples is the class of *linear magmas*, where the carrier M is a ring (which may be commutative or non-commutative, finite or infinite), and the operation $\diamond$ is given by $x \diamond y = ax + by$ for some coefficients $a, b \in M$; one can also generalize this slightly to *affine magmas*, in which the operation is given by $x \diamond y = ax + by + c$, but for simplicity we shall focus on linear magmas here. It is easy to see that in a linear magma, any word $w(x_1, \dots, x_n)$ of n indeterminates also takes the linear form

$$w(x_1, \dots, x_n) = \sum_{i=1}^{n} P_{w,i}(a, b) x_i$$

for some (possibly non-commutative) polynomial $P_{w,i}$ in a, b with integer coefficients. Thus, a linear magma will obey an equational law $w_1 \simeq w_2$ if and only if the pair (a, b) lies in the (possibly non-commutative) variety

$$\{(a, b) \in M \times M : P_{w_1,i}(a, b) = P_{w_2,i}(a, b) \text{ for all } i\}. \tag{3}$$

As such, a necessary condition for such a law $w_1 \simeq w_2$ to entail another law $w_1' \simeq w_2'$ is that one has the inclusion

$$\{(a, b) \in M \times M : P_{w_1,i}(a, b) = P_{w_2,i}(a, b) \text{ for all } i\} \subset \{(a, b) \in M \times M : P_{w_1',i}(a, b) = P_{w_2',i}(a, b) \text{ for all } i\}$$

for all rings M. For commutative rings, this criterion can be checked by standard Grobner basis techniques; in the noncommutative case one can use methods such as the diamond lemma [5].

Example 5.1 (Commutative counterexample)**.** For the law $x = y \diamond (((x \diamond y) \diamond x) \diamond y)$ (E1286), the variety Equation (3) can be computed to be

$$\{(a,b) \in M \times M : 1 = a + ba^3 + bab, 0 = a + ba^2b + b^2\}$$

while the variety for the idempotent law (E3) is

$$\{(a,b) \in M : a + b = 1\}.$$

Thus, to show that (E1286) does not entail (E3), it suffices to locate elements a, b of a ring M for one has $1 = a+ba^3+bab$, $0 = a+ba^2b+b^2$, and $a+b \neq 1$. Here one can take a commutative example, for instance when $M = \mathbb{Z}/p\mathbb{Z}$ and $(p, a, b) = (11, 1, 7)$ or $(p, a, b) = (7, 6, 2)$.

Example 5.2 (Noncommutative counterexample)**.** For the law $x = y \diamond ((y \diamond (x \diamond z)) \diamond z)$ (E1117), the variety Equation (3) can be computed to be

$$\{(a,b) \in M \times M : 1 = baba, 0 = a + ba^2, 0 = bab^2 + b^2\}$$

while the variety for $x = (x \diamond ((x \diamond x) \diamond x)) \diamond x$ (E2441) is

$$\{(a,b) \in M \times M : a^2 + aba^2 + abab + ab^2 + b = 1\}.$$

Observe that if $ba = -1$, then (a, b) automatically lies in the first set, and lies in the second set if and only if $(ab+1)(b-1) = 0$. One can then show that (E1117) does not imply (E2441) by setting $a = L$, $b = -R$ where L, R are the left and right shift operators respectively on the ring of integer-valued sequences $\mathbb{Z}^{\mathbb{N}}$. With some *ad hoc* effort one can convert this example into a less linear, but simpler (and easier to formalize) example, namely the magma with carrier $\mathbb{Z}$ and operation $x \diamond y = 2x - \lfloor y/2 \rfloor$.

Remark 5.3. As essentially observed in [1], if there is a commutative linear counterexample to an implication $E \vdash E'$, then by the Lefschetz principle this counterexample can be realized in a finite field $\mathbb{F}_q$ for some prime power q (and by the Chebotarev density theorem one can in fact take q to be a prime, so that the carrier is of the form $\mathbb{Z}/p\mathbb{Z}$ for some prime p), so that one also has $E \vdash_{\text{fin}} E'$. As such, we have found that an effective way to refute implications by the commutative linear magma method is to simply perform a brute force search over linear magmas $x \diamond y = ax + by$ in $\mathbb{Z}/p\mathbb{Z}$ for various triples (p, a, b). **Discuss performance of this method.**

On the other hand, the refutations obtained by non-commutative linear constructions need not have a finite model. For instance, consider the refutation $E1117 \not\vdash E2441$ from Example 5.2. The law (E1117) can be rewritten as $L_y R_z L_y R_z x = x$. This implies that R_z is injective and L_y is surjective for all y, z. For finite magmas $\mathcal{M}$, this then implies that the L_y, R_z are in fact invertible, and hence we have also $R_z L_y R_z L_y x = x$, which implies (E2441) by setting $x = y = z$. Thus, the refutation $E1117 \not\vdash E2441$ is "immune" to finite counterexamples.

Remark 5.4. One can also consider nonlinear magma models, such as quadratic models $x \diamond y = ax^2 + bxy + cy^2 + dx + ey + f$ in a cyclic group $\mathbb{Z}/N\mathbb{Z}$. For small values of N, we have found such models somewhat useful in providing additional refutations of implications $E \vdash_{\text{fin}} E'$ beyond what can be achieved by the linear or affine models. However, as the polynomials associated to a word $w(x_1, \ldots, x_n)$ tend to be of high degree (exponential in the order of the word), it becomes quite rare for such models to obey a given equation E when N is large.

Remark 5.5. One can also consider the seemingly more general linear model $x \diamond y = ax + by$, where the carrier M is now an abelian group, and a, b act on M by homomorphisms, that is to say that they are elements of the endomorphism ring $\mathrm{End}(M)$. However, this leads to exactly the same varieties Equation (3) (where M is now replaced by the endomorphism ring $\mathrm{End}(M)$) and so does not increase the power of the linear model for the purposes of refuting implications.

Give some statistics of what proportion of refutations can be resolved by linear models.

On the other hand, there are certainly some refutations $E \not\vdash E'$ of implications that are "immune" to both commutative and non-commutative models, in the sense that all such models that obey E, also obey E'. One such example is the refutation $E1485 \vdash E151$, which we discuss further in Section 5.4 below.

5.3. **Translation-invariant models.** It is natural to look for counterexamples amongst magmas that obey a large number of symmetries. One such class of counterexamples are *translation-invariant models*, in which the carrier M is a group, and the left translations of this group form isomorphisms of the magma M. In the case of an abelian group $M = (M, +)$, such models take the form

$$x \diamond y = x + f(y - x) \tag{4}$$

for some function $f\colon M \to M$; in the case of a non-abelian group $M = (M, \cdot)$, such models instead take the form

$$x \diamond y = x f(x^{-1} y). \tag{5}$$

For such models, the verification of an equational law in n variables corresponds to a functional equation for f in $n-1$ variables, as the translation symmetry allows one to normalize one variable to be the identity (say). This can simplify an implication to the point where an explicit counterexample can be found. These functional equations are trivial to analyze when $n = 1$. For $n = 2$, they are not as trivial, but still quite tractable, and has led to several refutations in practice. The method does not appear to be particularly effective for $n > 2$ due to the complexity of the functional equations.

Example 5.6 (Abelian example)**.** For the law $\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{y}) \diamond ((\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{y})$ (E1648), we apply the abelian translation-invariant model Equation (4) with $y = x + h$ to obtain

$$\begin{aligned} x \diamond y &= x + f(h) \\ (x \diamond y) \diamond y &= x + f(h) + f(h - f(h)) \\ (x \diamond y) \diamond ((x \diamond y) \diamond y) &= x + f(h) + f(f(h - f(h))) \end{aligned}$$

so that this law obeys (E1648) if and only if the functional equation

$$f(h) + f(f(h - f(h))) = 0$$

holds for all $h \in M$. Similarly, the law $\mathrm{x} \simeq (\mathrm{x} \diamond (\mathrm{x} \diamond \mathrm{y})) \diamond \mathrm{y}$ (E206) is obeyed if and only if

$$f(f(h)) + f(h - f(f(h))) = 0$$

for all $h \in M$. One can now check that the function $f\colon \mathbb{Z} \to \mathbb{Z}$ defined by $f(h) := -\mathrm{sgn}(h)$ (thus $f(h)$ equals -1 when h is positive, $+1$ when h is negative, and 0 when h is zero) obeys the first functional equation but not the second, thus establishing that $E1648 \not\models E206$.

Example 5.7 (Non-abelian example)**.** We now obtain the opposite refutation $E206 \not\models E1648$ to Example 5.6 using the non-abelian translation-invariant model. By similar calculations to before, we now seek to find a function $f\colon M \to M$ on a non-abelian group $(M, \cdot)$ that obeys the functional equation

$$f(f(h))f(f(f(h))^{-1}h) = 1 \tag{6}$$

for all $h \in M$, but fails to obey the functional equation

$$f(h)f(f(f(h)^{-1}h)) = 1 \tag{7}$$

for at least one $h \in M$. Now take M to be the group generated by three generators a, b, c subject to the relations $a^2 = b^2 = c^2 = 1$, or equivalently the group of reduced words in a, b, c with no adjacent letters in the word equal. We define

$$f(1) = 1, f(a) = b, f(b) = c, f(c) = a$$

and then $f(aw) = a$ for any non-empty reduced word w not starting with a, and similarly for b and c. The equation (6) can be checked directly for $h = 1, a, b, c$. If $h = aw$ with w non-empty, reduced, and not starting with a, then $f(f(h))^{-1} = f(f(h)) = b$ and $f(f(f(h))^{-1}h) = f(baw) = b$, giving (6) in this case, and similarly for cyclic permutations. Meanwhile, (7) can be checked to fail for $h = a$.

Remark 5.8. The construction in Example 5.7 also has the following more "geometric" interpretation. The carrier M can be viewed as the infinite 3-regular tree, in which every vertex imposes a cyclic ordering on its 3 neighbors (for instance, if we embed M as a planar graph, we can use the clockwise ordering). For $x, y \in M$, we then define $x \diamond y$ to equal x if $x = y$. If y is instead a neighbor of x, we define $x \diamond y$ to be the next neighbor of x in the cyclic ordering. Finally, if y is distance two or more from x, we define $x \diamond y$ to be the neighbor of x that is closest to y. One can then check that this model obeys (6) but not (7).

Remark 5.9. These constructions are necessarily infinitary in nature, because (E206) and (E1648) can be shown to be equivalent for finite magmas. Indeed, (E206) can be written as $x = R_y L_x L_x y$, which implies that R_y is surjective, hence injective, on a finite magma; writing $x = R_y z$ we conclude that $R_y z = R_y L_{z\diamond y} L_{z\diamond y} y$ and hence $z = L_{z\diamond y} L_{z\diamond y} y$, giving (E1648). The opposite implication is similar (using (E1648) to show that R_y is injective, hence surjective), and is left to the reader.

Some refutations $E \not\models E'$ are "immune" by translation-invariant models, in that any translation-invariant model that obeys E, also obeys E'. One obstruction is that for such models, the squaring map S is necessarily an invertible map, since $Sx = x + f(0)$ in the abelian case and $Sx = xf(1)$ in the non-abelian case. On the other hand, adding the assumption of invertibility of squares can sometimes make the implication $E \vdash E'$. For instance, the commutative law $\mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{y}) \simeq (\mathrm{y} \diamond \mathrm{y}) \diamond \mathrm{x}$ (E4482) for a square and an arbitrary element will imply the full commutative law (E43) for translation-invariant models due to the surjectivity of S, but does not imply it in general (as one can easily see by considering models where S is constant).

5.4. **The twisting semigroup.** Suppose one has a magma $\mathcal{M}$ obeying a law E, that also enjoys some endomorphisms $T, U\colon \mathcal{M} \to \mathcal{M}$. Then one can "twist" the operation $\diamond$ by T, U to obtain a new magma operation

$$x \diamond' y := Tx \diamond Uy. \tag{8}$$

If one then tests whether this new operation $\diamond'$ obeys the same law E as the original operation $\diamond$, one will find that this will be the case provided that T, U obey a certain set of relations. The semigroup generated by formal generators T, U with these relations will be called the *twisting semigroup* Twist_E of E. This can be best illustrated with some examples.

Example 5.10. We compute the twisting semigroup of $\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond (\mathrm{z} \diamond \mathrm{y}))$ (E1485). We test this law on the operation Equation (8), thus we consider whether

$$x = (y \diamond' x) \diamond' (x \diamond' (z \diamond' y))$$

holds for all $x, y, z \in M$. Substituting in Equation (8) and using the homomorphism property repeatedly, this reduces to

$$x = (T^2 y \diamond TUx) \diamond (UTx \diamond (U^2 Tz \diamond U^3 y)).$$

If we impose the conditions $TU = UT$, $T^2 = U^3$, then this equation would follow from (E1485) (with x, y, z replaced with TUx, $T^2 y$, $U^2 Tz$ respectively). Thus the twisting semigroup Twist_{E1485} of (E1485) is generated by two generators T, U subject to the relations $\mathrm{TU} = \mathrm{UT} = 1$, $\mathrm{T}^2 = \mathrm{U}^3$. This is a cyclic group of order 5, since the relations can be rewritten as $\mathrm{T}^5 = 1$, $\mathrm{U} = \mathrm{T}^{-1}$.

Now consider $\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond \mathrm{x})$ (E151). Applying the same procedure, we arrive at

$$x = (T^2 x \diamond TUx) \diamond (UTx \diamond U^2 x)$$

so the twisting group Twist_{E151} is generated by two generators T, U subject to the relations $\mathrm{TU} = \mathrm{UT} = \mathrm{T}^2 = \mathrm{U}^2 = 1$. This is a cyclic group of order 2, since the relations can be rewritten as $\mathrm{T}^2 = 1$, $\mathrm{U} = \mathrm{T}$.

Suppose the twisting semigroup Twist_E is not a quotient of $\mathrm{Twist}_{E'}$, in the sense that the relations that define $\mathrm{Twist}_{E'}$ are not obeyed by the generators of Twist_E. Then one can often disprove the implication $E \vdash E'$ by attempting the following procedure.

- First, locate a non-trivial magma $\mathcal{M} = (M, \diamond)$ obeying the law E. Then the Cartesian power M^{Twist_E} of tuples $(x_W)_{W \in \mathrm{Twist}_E}$, with the pointwise magma operation, will also obey E.
- Furthermore, this Cartesian power admits two endomorphisms T, U defined by

  $$T(x_W)_{W \in \mathrm{Twist}_E} = (x_{W\mathrm{T}})_{W \in \mathrm{Twist}_E}; U(x_W)_{W \in \mathrm{Twist}_E} = (x_{W\mathrm{U}})_{W \in \mathrm{Twist}_E},$$

  which obey the relations defining Twist_E.
- We now twist the magma operation $\diamond$ on M^{Twist_E} by T, U to obtain a new magma operation $\diamond'$ defined by Equation (8), that will still obey law E.
- Because T, U will not obey the relations defining $\mathrm{Twist}_{E'}$, it is highly likely that this twisted operation will not obey E', thus refuting the implication $E \vdash E'$. If M and the twisting semigroup were finite, this approach should also refute $E \vdash_{\mathrm{fin}} E'$.

For instance, a non-trivial finite model for (E1485) is given by the finite field $\mathbb{F}_2$ of two elements with the NAND operation $x \diamond y := 1 - xy$. If we twist $\mathbb{F}_2^5$ by the left shift $T(x_i)_{i=1}^5 = (x_{i+1})_{i=1}^5$ and right shift $U(x_i)_{i=1}^5 = (x_{i-1})_{i=1}^5$, where we extend the indices periodically modulo 5, then the resulting operation

$$(x_i)_{i=1}^5 \diamond' (y_i)_{i=1}^5 := (1 - x_{i+1} y_{i-1})_{i=1}^5$$

on $\mathbb{F}_2^5$ will still obey (E1485), but will not obey (E151), thus showing that $E1485 \not\models_{\text{fin}} E151$ and hence $E1485 \not\models E151$. This particular implication does not seem to be easily establishable by any of the other methods discussed in this paper.

Report on how large the twisting semigroups are in practice, and how many implications can be refuted by this method.

5.5. **Greedy constructions.** We have found *greedy extension methods*, or *greedy methods* for short, are a powerful way to refute implications, especially when the carrier M is allowed to be infinite. A basic implementation of this method is as follows. To build a magma operation $\diamond \colon M \times M \to M$ that obeys one law E but not another E', one can first consider *partial magma operations* $\diamond \colon \Omega \to M$, defined on some subset Ω of $M \times M$. Thus $x \diamond y$ is defined if and only if $(x, y) \in \Omega$. A magma operation is then simply a partial operation which is *total* in the sense that $\Omega = M \times M$. We say that a partial magma operation is *finitely supported* if Ω is finite.

In the language of first-order logic, a partial magma operation can also be viewed as a ternary relation $R(x, y, z)$ on M with the axiom that $R(x, y, z) \wedge R(x, y, z') \implies z = z'$ for all $x, y, z \in M$. The support Ω is then the set of (x, y) for which $R(x, y, z)$ holds for some (necessarily unique) z, which one can then take to be the definition of $z = x \diamond y$.

We say that one partial operation $\diamond' \colon \Omega' \to M$ *extends* another $\diamond \colon \Omega \to M$ if Ω' contains Ω, and $x \diamond y = x \diamond' y$ whenever $x \diamond y$ (and hence $x \diamond' y$) are defined. Given a sequence $\diamond_n \colon \Omega_n \to M$ of partial operations, each of which is an extension of the previous, we can define the *direct limit* $\diamond_\infty \colon \bigcup_n \Omega_n \to M$ to be the partial operation defined by $x \diamond_\infty y := x \diamond_n y$ whenever $(x, y) \in \Omega_n$.

Abstractly, the greedy algorithm strategy can now be described as follows.

Theorem 5.11 (Abstract greedy algorithm)**.** *Let E, E' be equational laws, and let Γ be a theory of first-order sentences regarding a partial magma operations $\diamond \colon \Omega \to M$ on a carrier M. Assume the following axioms:*

- *(i) (Seed) There exists a finitely supported partial magma operation $\diamond_0 \colon \Omega_0 \to M$ satisfying Γ that contradicts E', in the sense that there is some assignment of variables in E' in M such that both sides of E' are defined using $\diamond_0$, but not equal to each other.*
- *(ii) (Soundness) If $\diamond_n \colon \Omega_n \to M$ is a sequence of partial magma operations obeying Γ with each $\diamond_{n+1}$ an extension of $\diamond_n$, and the direct limit $\diamond_\infty$ is total, then this limit obeys E.*
- *(iii) (Greedy extension) If $\diamond \colon \Omega \to M$ is a finitely supported partial magma operation obeying Γ, and $a, b \in M$, then there exists a finitely supported extension $\diamond' \colon \Omega' \to M'$ of $\diamond$ to a possibly larger carrier M' such that $a \diamond' b$ is defined.*

Then $E \not\vdash E'$.

We remark that this greedy method seems to be inherently infinitary in nature, and does not seem well adapted to refute finite magma implications $E \vdash_{\rm fin} E'$.

Proof. We work on the countably infinite carrier $\mathbb{N}$. By embedding the finitely supported operation $\diamond_0$ from axiom (i) into $\mathbb{N}$, we can assume without loss of generality that $\diamond_0$ has carrier $\mathbb{N}$. By similar relabeling, we can assume in (iii) that $M' = M$ when $M = \mathbb{N}$, since any elements of $M' \backslash \mathbb{N}$ that appear in Ω' can simply be reassigned to natural numbers that did not previously appear in Ω. We well-order the pairs in $\mathbb{N} \times \mathbb{N}$ by (a_n, b_n) for $n = 1, 2, \dots$. Iterating (iii) starting from $\diamond_0$, we can thus create a sequence of finitely supported magma operations $\diamond_0, \diamond_1, \dots$ on $\mathbb{N}$ obeying Γ, with each $\diamond_{n+1}$ an extension of $\diamond_n$, and $a_n \diamond_n b_n$ defined for all $n \geq 1$. Then the direct limit $\diamond_\infty$ of these operations is total, and does not obey E' thanks to axiom (i). On the other hand, by axiom (ii) it obeys E, and the claim follows. □

We refer to Γ as the *rule set* for the greedy extension method. To apply Theorem 5.11 to obtain a refutation $E \vdash E'$, we have found the following trial-and-error method to work well in practice:

1. Start with a minimal rule set Γ that has just enough axioms to imply the soundness property for the given hypothesis E.
2. Attempt to establish the greedy extension property for this rule set by setting $a \diamond' b$ equal to a new element $c \notin M$, and then defining additional values of $\diamond'$ as necessary to recover the axioms of Γ'.
3. If this can be done in all cases, then locate a seed $\diamond_0$ refuting the given target E', and STOP.
4. If there is an obstruction (often due to a "collision" in which a given operation $x \diamond' y$ is required to equal two different values), add one or more rules to Γ to avoid this obstruction, and return to Step 2.

As an example, we present

Proposition 5.12 (73 does not imply 4380)**.** *The law* $\mathrm{x} \simeq \mathrm{y} \diamond (\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{y}))$ (E73) *does not imply* $\mathrm{x} \diamond (\mathrm{x} \diamond \mathrm{x}) \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond \mathrm{x}$ (E4380).

Proof. To build a rule set Γ that will imply (E73) when total, a natural first choice would be the single rule

1. If $y \diamond (x \diamond y)$ is defined, then $y \diamond (y \diamond (x \diamond y))$ is defined and equal to x.

However, the greedy algorithm will fail just with this rule: if the partial operation has $x \diamond y$ and $z \diamond y$ both equal to some w for some $x \neq z$, then any attempt to assign a value to $y \diamond w$ will lead to a contradiction, as the above rule will force $y \diamond w$ to equal both x and z. Indeed, it is clear that (E73) forces all the right translation operators R_y to be injective. We therefore add this as an additional rule:

2. If $x \diamond y$ and $z \diamond y$ are defined and equal, then $x = z$.

To avoid some unwanted edge cases, it is also convenient to impose the additional rule

3. If $x \diamond y$ is defined, it is not equal to y.

Unlike Rule 2, this rule is not forced by (E73), but can be enforced as part of the greedy construction.

The ruleset clearly obeys the soundness axiom (ii) of Theorem 5.11. We now verify the greedy extension axiom (iii). Let Ω, a, b be as in that axiom. We may assume that $a \diamond b$ is undefined, since we are done otherwise. We adjoin a new element c to M to create M', and set $a \diamond' b = c$. If we also have $b = d \diamond a$ for some d (unique by Rule 2, and only possible for $a \neq b$ by Rule 3), set $a \diamond' c = d$ (this is necessary to retain Rule 1). Of course, we also set $x \diamond' y = x \diamond y$ whenever $x \diamond y$ is already defined.

Since $c \notin M$, it is clear that $\diamond'$ is a finitely supported partial magma operation on M'. It is also clear that $\diamond'$ obeys Rule 2 and Rule 3. Now we case check Rule 1:

- Case 1: $x = c$ or $y = c$. Not possible since no left multiplication with c is defined.
- Case 2: $x \diamond' y = c$. Only possible when $x = a$, $y = b$, but then $y \diamond' (x \diamond' y)$ is undefined since $y = b \neq a$ if d is defined.
- Case 3. $y \diamond' (x \diamond' y) = c$. Only possible when $y = a$ and $x = d$, and holds in this case.
- Case 4: $x, y, x \diamond' y, y \diamond' (x \diamond' y) \neq c$: this case is covered by Rule 3 for $\diamond$.

To conclude, we need to locate a seed $\diamond_0$ obeying Rules 1,2,3 but contains a counterexample to (E4380). One simple example is the carrier $\{0, 1, 2, 3\}$ with $0 \diamond_0 0 = 1$, $0 \diamond_0 1 = 2$, $0 \diamond_0 2 = 0$, $1 \diamond_0 0 = 3$. □

This method is not guaranteed to halt in finite time, as there may be increasingly lengthy sets of rules one has to add to Γ to avoid collisions. However, in practice we have found many of the refutations that could not be resolved by simpler techniques to be amenable to this method (or variants thereof, as discussed below).

One can automate the above procedure by using ATPs (or SAT solvers) to locate new rules that are necessary and sufficient resolve any potential collision (and which, *a posteriori*, can be seen to be necessarily consequences of the law E). The seed-finding step (Step 3) is particularly easy to automate, and can also often be done by hand. **Describe performance of this automated method. Discuss the issue that some implications required a large SAT solver calculation that was difficult to formalize efficiently in Lean, prompting human-generated simplified proofs using smaller rulesets.**

However, in some cases we have found it necessary to add "inspired" choices of rules that were not forced by the initial hypothesis E, but which simplified the analysis by removing problematic classes of collisions from consideration. We were unable to fully automate the process of guessing such choices; however, we found ATPs very useful for testing any proposed such guess. In particular, if an ATP was able to show that the existing ruleset, together with a proposed new rule A, implied E', then this clearly indicated that one should not add A to the rule set Γ. Conversely, if an ATP failed to establish such an implication, this was evidence that this was a "safe" rule to impose.

We also found that human verification of the greedy extension property was a highly error-prone process, as the case analysis often included many delicate edge cases that were easy to overlook. Both ATPs and the Lean formalization therefore played a crucial role in verifying the human-written greedy arguments, often revealing important gaps in those arguments that required either minor or major revisions to the rule set.

The greedy method can also be combined with the translation-invariant method, both in abelian and non-abelian settings. For instance, we can modify the proof of Theorem 5.11 to obtain the following variant:

Theorem 5.13 (Non-commutative translation-invariant greedy algorithm). *Let F, F' be functional equations on groups, and let Γ be a theory of first-order sentences regarding a partial function $f\colon \Omega \to G$ on a group $G = (G, \cdot)$. Assume the following axioms:*

(i) (Seed) There exists a finitely supported partial function $f_0\colon \Omega_0 \to G$ satisfying Γ that contradicts F', in the sense that there is some assignment of variables in F' in G such that both sides of F' are defined using f_0, but not equal to each other.
(ii) (Soundness) If $f_n\colon \Omega_n \to G$ is a sequence of partial functions obeying Γ with each f_{n+1} an extension of f_n, and the direct limit f_∞ is total, then this limit obeys F.
(iii) (Greedy extension) If $f\colon \Omega \to G$ is a finitely supported partial function obeying Γ, and $a, b \in G$, then there exists a finitely supported extension $f'\colon \Omega' \to G'$ of f to a possibly larger group G' such that $a \diamond' b$ is defined.

Then $F \not\vdash F'$.

One can of course also develop an abelian analogue of the above theorem, in which $G = (G, +)$ and $G' = (G', +)$ are now required to be abelian. We can then give an alternate proof of Proposition 5.12 as follows:

Second proof of Proposition 5.12. (Sketch) The functional equations associated to (E73) and (E4380) are $f^2(h^{-1}f(h)) = h^{-1}$ and $f^2(1) = f(1)f(f(1)^{-1})$ respectively. We apply Theorem 5.13 with the following ruleset:

1. If $f(h^{-1}f(h))$ is defined, then $f^2(h^{-1}f(h))$ is defined and equal to h^{-1}.
2. If $h^{-1}f(h)$ and $k^{-1}f(k)$ are defined and equal, then $h = k$.
3. If $f(h)$ is defined, it is not equal to h.

Axiom (ii) is clear. To verify axiom (iii), we can assume $f(h)$ is undefined, then adjoin an element c freely to G to create a larger group G', and set $f'(h) = c$. If $h = k^{-1}f(k)$ for some k (which is unique by Rule 2, and only possible for $h \neq 1$ by Rule 3), then also set $f'(c) = k^{-1}$. One can then check that axiom (iii) is obeyed. For axiom (i), take G to be a free cyclic group with one generator a, and set $f(1) = a$, $f(a) = a^3$, $f(a^3) = 1$, $f(a^{-1}) = a^3$ (say). ∎

More complex (and *ad hoc*) variants of the greedy algorithm are possible. In some cases, we were not able to preserve the finitely supported nature of the partial operation or partial function, and needed to extend that partial object at an infinite number of values at each step. In other cases, one also had to add additional temporary data during the greedy process to record tasks that one wished to attend to at a later stage of the process, but could not handle immediately because it was awaiting some other operation to become well-defined. We will not attempt to survey all possible variants of this method here, but refer the reader to the ETP blueprint for further examples.

5.6. **Modifying base models.** A general technique that we have found useful in obtaining a refutation such as $E \not\models E'$ is to start with a simple base model $\mathcal{M} = (M, \diamond)$ that obeys both E and E', and modify it in various ways to preserve E, but create a violation of E'. There are many such possible modifications, but three general ways that have proven effective are as follows:

(i) Modify the magma operation $\diamond\colon M \times M \to M$ on a small set in order to violate E', and then make further modifications as needed to recover E.
(ii) Construct an *extension* $\mathcal{N}$ of $\mathcal{M}$, equipped with a surjective magma homomorphism $\pi : \mathcal{N} \to \mathcal{M}$, and defined in terms of some additional data. Then solve for that data in such a way that N obeys E but not E'.
(iii) Construct an *enlargement* $\mathcal{M}' = (M', \diamond')$ of $\mathcal{M} = (M, \diamond)$, which is a magma that contains $\mathcal{M}$ as a submagma. One needs to construct the multiplication table $\diamond$ on $(M' \times M') \backslash (M \times M)$ in order to retain E but disprove E'.

One appealing case of (ii), involving a "magma cohomology" analogous to (abelian) group cohomology, is that of an *affine* extension of a magma $\mathcal{G} = (G, \diamond_G)$ by another magma $(M, \diamond_M)$ which is an abelian group M with a linear magma operation $s \diamond_M t := as + bt$ for some endomorphisms $a, b \in \mathrm{End}(M)$. One can then consider extensions with carrier $G \times M$ and magma operation

$$(x, s) \diamond (y, t) := (x \diamond_G y, s \diamond_M t + f(x, y)) \tag{9}$$

for some function $f\colon G \times G \to M$. If $(M, \diamond_M)$ and $(G, \diamond_G)$ already obey a law E, then this extension will also obey E if and only if f obeys a certain "cocycle equation", which is a linear equation on f. One can then sometimes use linear algebra to locate an f that obeys the cocycle equation for one law E but not another E', thus refuting the implication $E \vdash E'$. An example is as follows:

Proposition 5.14 (1110 does not imply 1629)**.** *The law* $\mathrm{x} \simeq \mathrm{y} \diamond ((\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{x})) \diamond \mathrm{y})$ (E1110) *does not imply* $\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond ((\mathrm{x} \diamond \mathrm{x}) \diamond \mathrm{x})$ (E1629)*, even for finite magmas.*

Proof. (Sketch) Using the linear ansatz, we find that (E1110) has a model $\mathcal{M}$ with carrier $\mathbb{F}_5$ (the finite field $\mathbb{Z}/5\mathbb{Z}$) with operation $x \diamond y = 3x - y$. We then apply the ansatz (9) with $G = M$. One then finds that this operation obeys (E1110) if $f\colon \mathbb{F}_5 \times \mathbb{F}_5 \to \mathbb{F}_5$ obeys the cocycle equation

$$3f(x, x) - 3f(y, 2x) - f(3y - 2x, y) + f(y, 3y - x) = 0$$

for all $x, y \in \mathbb{F}_5$, and obeys (E1629) if f obeys the cocycle equation

$$f(2x, 0) - f(2x, 2x) = 0$$

for all $x \in \mathbb{F}_5$. A routine symbolic algebra package computation reveals that the space of f that obeys the former equation is a six-dimensional vector space over $\mathbb{F}_5$, which is not contained in the solution space of the latter equation, giving the claim. In fact, since these equations preserve the space of homogeneous polynomials of a fixed degree, one can use linear algebra to locate an example that is a homogeneous polynomial; one explicit choice is

$$f(x, y) = y^5 + xy^4 + x^2y^3 + 3x^3y^2 + 3x^4y_1$$

. □

It may be of interest to develop this theory of "magma cohomology" further, for instance by defining higher order magma cohomology groups.

Now we give an example of how method (ii) can be combined with method (i).

Proposition 5.15 (1659 does not imply 4315)**.** $\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{y}) \diamond ((\mathrm{y} \diamond \mathrm{y}) \diamond \mathrm{z})$ (E1659) *does not imply* $\mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{x}) \simeq \mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z})$ (E4315)*.*

Proof. There are two simple models for (E1659): the model G with carrier $\mathbb{Z}/2\mathbb{Z}$ and operation $x \diamond y = x + 1$, and the model $\mathcal{M}$ with carrier $\mathbb{Z}$ and operation $x \diamond y = x$. Using the ansatz (9), one can soon discover that one obtains a magma operation $\diamond : (G \times M) \times (G \times M) \to G \times M$ with $f(0, 0) = f(1, 0) = 0$, $f(0, 1) = -1$, and $f(1, 1) = 1$. This model still obeys (E4315). However, we can create a modification $\diamond'$ of $\diamond$ as follows. We will seek to violate (E4315) at $x = (0, 0)$, $y = (0, 0)$, $z = (1, 0)$, thus we want

$$(0, 0) \diamond' ((0, 0) \diamond' (0, 0)) \neq (0, 0) \diamond' ((0, 0) \diamond' (1, 0)).$$

We have $(0, 0) \diamond (0, 0) = (1, 0)$ and $(0, 0) \diamond (1, 0) = (1, -1)$. One can try to force the counterexample by setting $(0, 0) \diamond' (1, 0)$ to equal $(0, 0)$ instead of $(1, -1)$. However, if this is the only change we make, then we no longer obey (E1659), since

$$(1, 0) \neq ((0, 0) \diamond' (1, 0)) \diamond' (((1, 0) \diamond' (1, 0)) \diamond (1, t))$$

for any $t \in \mathbb{Z}\backslash\{0\}$. But these are the only counterexamples created; and if one then sets $(0, 0) \diamond' (1, t) = (0, 0)$ for *all* $t \in \mathbb{Z}$, then one can check that the modified operation $\diamond'$ now obeys (E1659) but not (E4315) as required. □

Finite models for the law $\mathrm{x} \simeq \mathrm{x} \diamond ((\mathrm{y} \diamond \mathrm{z}) \diamond (\mathrm{x} \diamond \mathrm{z}))$ (E854) seems to be somewhat "mutable" in that one can often change a small number of entries in the multiplication table, and also add an additional row and column to the table, in ways that preserve the law (E854). This renders this law suitable for using methods (i), (iii) to construct new models of this equation that refute various implications $E854 \vdash_{\mathrm{fin}} E$, for instance by starting with a model that already refuted some stronger law E', and attempt to modify it (possibly with ATP assistance) by some combination of methods (i), (iii) to then violate E. **Explain this in more detail**

Another way to utilize (iii), which proved useful for laws that involved the squaring operator S, was to adopt a "squares first" approach in which one selected a base model $S\mathcal{M} = (SM, \diamond)$

to serve as the set of squares, then extend it to a larger model $\mathcal{M}$ with carrier $M = SM \uplus N$ by first determining what the multiplication map should be on the diagonal $\{(x,x) : x \in N\}$ (i.e., to determine the squaring map $S \colon N \to SM$), together with the values on the blocks $SM \times N$, $N \times SM$, and then finally resolve the remaining values on the $N \times N$ block. Often, versions of the greedy algorithm are useful for each of these stages of the construction. The precise details are quite technical, particularly for the law $\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{y}) \diamond ((\mathrm{y} \diamond \mathrm{x}) \diamond \mathrm{y})$ (E1729), which was the last of the equations whose implications were settled by the ETP. We refer the reader to the ETP blueprint for further details.

6. Syntactic arguments

Many proofs or refutations of implications (or equivalences) between two equational laws E, E' can be obtained from the syntactic form of the equation. We discuss some techniques here that were useful in the ETP.

6.1. **Simple rewrites.**

Many equational laws E' can be formally deduced from a given law E by applying the Lean 'rw' tactic to rewrite E' repeatedly by some forward or backward application of E applied to arguments that match some portion of E. For instance, the commutative law (E43) clearly implies $\mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z}) \simeq (\mathrm{y} \diamond \mathrm{z}) \diamond \mathrm{x}$ (E4531) by a single such rewrite. A brute force application of such rewrite methods is already able to directly generate about $15,000$ such implications, including many equivalences to the singleton law (E2) and the constant law (E46). After applying transitive closure, this generates about four million further such implications.

A simple observation that already generates many equivalences is that any equation of the form $\mathrm{x} \simeq f(\mathrm{y}, \mathrm{z}, \dots)$ necessarily is equivalent to the trivial law $\mathrm{x} \simeq \mathrm{y}$; similarly, an equation of the form $f(\mathrm{x}, \mathrm{y}) \simeq g(\mathrm{z}, \mathrm{w}, \dots)$ implies $f(\mathrm{x}, \mathrm{y}) \simeq f(\mathrm{x}', \mathrm{y}')$; and so forth. **Give some stats on how effective this is.**

6.2. **Matching invariants.**

Fix an alphabet X. A *matching invariant* is an assignment $I \colon \mathcal{M}_X \to \mathcal{I}$ of an object $I(w) \in \mathcal{I}$ in some space $\mathcal{I}$ to each word $w \in \mathcal{M}_X$ with the property that if an equational law $w_1 \simeq w_2$ has matching invariants $I(w_1) = I(w_2)$, then the same matching $I(w_1') = I(w_2')$ holds for any consequence $w_1' \simeq w_2'$. In particular, if one law $I(w_1) = I(w_2)$ and $I(w_1') \neq I(w_2')$, then the law $w_1 \simeq w_2$ does not imply the law $w_1' \simeq w_2'$.

A simple example of a matching invariant is the multiplicity $(n_x)_{x \in X}$ of variables of a word: if w_1, w_2 have all variables x appear the same number of times n_x in both words, then any rewriting of a word w using the law $w_1 \simeq w_2$ will preserve this property. Hence, if w_1', w_2' do not have that each variable appear the same number of times in both words, then $w_1 \simeq w_2$ cannot imply $w_1' \simeq w_2'$. For instance, the commutative law (E43) cannot imply the left-absorptive law (E4).

One source of matching invariants comes from the free magma $\mathcal{M}_{X,\Gamma}$ of a theory:

Proposition 6.1 (Free magmas and matching invariants)**.** *Let Γ be a theory, and let $\iota_{X,\Gamma}\colon X \to \mathcal{M}_{X,\Gamma}$ be the map associated to the free magma $\mathcal{M}_{X,\Gamma}$ for that theory. Then the map $I\colon \mathcal{M}_X \to \mathcal{M}_{X,\Gamma}$ defined by $I(w) := \varphi_{\iota_{X,\Gamma}}(w)$ is an invariant.*

Proof. Suppose that $w_1 \simeq w_2$ entails $w_1' \simeq w_2'$, and that $I(w_1) = I(w_2)$. For any $f\colon X \to \mathcal{M}_{X,\Gamma}$, the two maps $\varphi_f, \varphi_{f,\Gamma} \circ \varphi_{\iota_{X,\Gamma}}\colon \mathcal{M}_X \to \mathcal{M}_{X,\Gamma}$ are both homomorphisms that extend f, hence agree by the universal property of $\mathcal{M}_X$, as displayed by the following commutative diagram:

$$\begin{tikzcd}
 & X \arrow[dl, hook'] \arrow[d, "\iota_{X,\Gamma}"'] \arrow[dr, "f"] & \\
\mathcal{M}_X \arrow[r, "I=\varphi_{\iota_{X,\Gamma}}"] \arrow[rr, bend right, "\varphi_f"'] & \mathcal{M}_{X,\Gamma} \arrow[r, "\varphi_{f,\Gamma}"] & \mathcal{M}_{X,\Gamma}
\end{tikzcd}$$

In particular, the hypothesis $I(w_1) = I(w_2)$ implies that $\varphi_f(w_1) = \varphi_f(w_2)$ for all $f\colon X \to \mathcal{M}_{X,\Gamma}$; that is to say, the magma $\mathcal{M}_{X,\Gamma}$ obeys the law $w_1 \simeq w_2$, and hence also $w_1' \simeq w_2'$ by hypothesis. In particular, $\varphi_{\iota_{X,\Gamma}}(w_1') = \varphi_{\iota_{X,\Gamma}}(w_2')$, which gives $I(w_1') = I(w_2')$ as required. $\square$

Example 6.2. If we take $\Gamma = \{E4\}$ to be the theory of the left-absorptive law (E4) as described in Example 2.2, then the matching invariant $I(w)$ produced by Proposition 6.1 is the left-most letter of the alphabet X appearing in the word; for instance $I((\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{z}) = \mathrm{x}$. Thus, for example, the left-absorptive law (E4) cannot imply the right-absorptive law (E5).

Example 6.3. If we take $\Gamma = \{E43, E4512\}$ to be the theory of the commutative law (E43) and the associative law (E4512), then by Example 2.1, the associated invariant $I(w) = \sum_{x\in X} n_x e_x$ is the formal sum of all the generators e_x appearing in the word w, in the free abelian semigroup generated by those generators. This recovers the preceding observation that the multiplicities $(n_x)_{x\in X}$ form a matching invariant.

Example 6.4. Let $n \geq 1$ be a positive integer, and consider the theory $\Gamma = \{E43, E4512, E_n\}$ consisting of the previous theory $\{E43, E4512\}$ together with the order-n law $L_x^y x = y$. One can check that the free magma $\mathcal{M}_{X,\Gamma}$ can be described as the free group of exponent n with generators $e_x, x \in X$, with associated map $\iota_{X,\Gamma}\colon x \mapsto e_x$. The associated matching invariant $I(w) = \sum_{x\in X} n_x e_x$ is essentially the multiplicities $(n_x \bmod n)_{x\in X}$ modulo n, which gives a slightly stronger criterion than the preceding matching invariant for refuting implications. For example, the cubic idempotent law $\mathrm{x} \simeq (\mathrm{x} \diamond x) \diamond \mathrm{x}$ (E23) has matching invariants $e_x = 3e_x$ in the $n = 2$ case, and hence does not imply the idempotent law $\mathrm{x} \simeq \mathrm{x} \diamond \mathrm{x}$ (E3) since $e_x \neq 2e_x$ in the $n = 2$ case.

Give some statistics on how many refutations can be established by these methods.

Remark 6.5. One can also obtain matching invariants from the free objects associated to theories that involve additional operations beyond the magma operation $\diamond$, such as an identity element or an inverse operation. We leave the precise generalization of Proposition 6.1 to such theories to the interested reader.

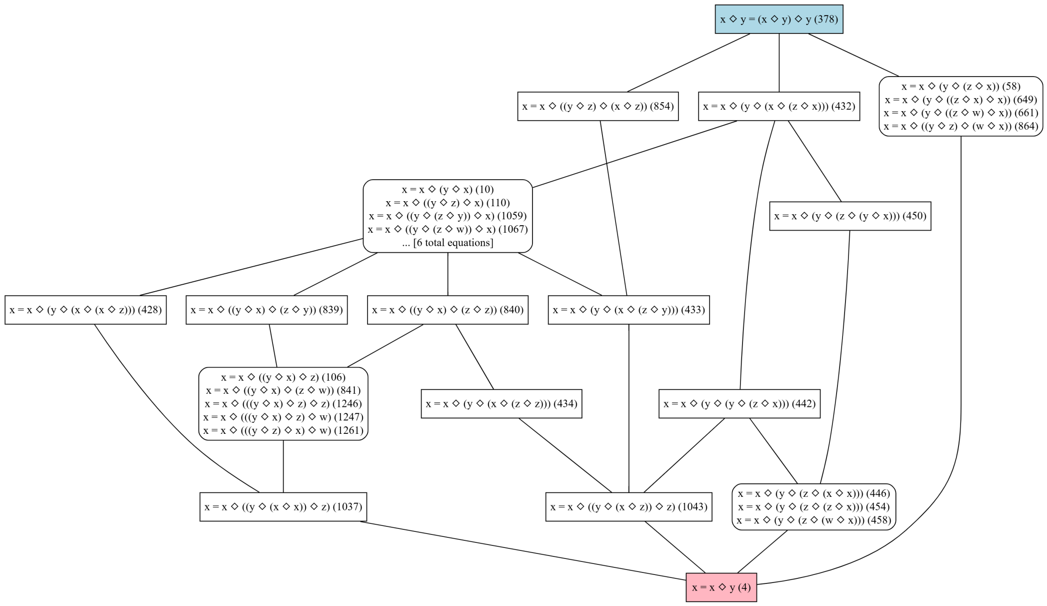

FIGURE 3. Equations similar to (E854) that are of the form Equation (10) (possibly involving a fourth indeterminate w) and imply (E378). For brevity, 70 equations equivalent to (E4) have been omitted.

6.3. **Confluence. Define a confluent law and give some examples.**

6.4. **Complete rewriting systems. Define a complete rewriting system and give some examples.**

6.5. **Unique factorization.** In general, the free magma $\mathcal{M}_{X,E}$ for a given equational law E, which we can canonically define as $\mathcal{M}_X/\sim_E$, is hard to describe explicitly; indeed, from the undecidability of implications between equational laws, such a magma cannot be computably described for arbitrary E. Nevertheless, for some laws it is possible to obtain some partial understanding of $\mathcal{M}_{X,E}$ from a syntactic perspective. For instance, if we can refute the equivalence $w_1' \sim_E w_2'$ by constructing a counterexample magma M that obeys E but not $w_1' \simeq w_2'$, then this implies that the representatives $\iota_{X,E}(w_1'), \iota_{X,E}(w_2')$ of w_1', w_2' in $\mathcal{M}_{X,E}$ are distinct.

We illustrate this approach with equations E of a left-absorptive form

$$\mathrm{x} \simeq \mathrm{x} \diamond f(\mathrm{x}, \mathrm{y}, \mathrm{z}) \tag{10}$$

for some word $f(\mathrm{x}, \mathrm{y}, \mathrm{z})$, which imply the right-idempotent law (E378).

An illustrative example is the law (E854) depicted in Figure 1. Other examples are listed in Figure 3.

Lemma 6.6. *Equation* (E854) *is of the form Equation* (10) *and implies* (E378).

Proof. Clearly we have Equation (10) with $f(\mathrm{x},\mathrm{y},\mathrm{z}) := (\mathrm{y} \diamond \mathrm{z}) \diamond (\mathrm{x} \diamond \mathrm{z})$. From Equation (10) we have in any magma obeying (E854) that

$$x = x \diamond f(x, S^2x, x) = x \diamond S(x \diamond S^2x) = x \diamond S(x \diamond f(x,x,x)) = x \diamond Sx.$$

This implies from a further application of Equation (10) that

$$y = y \diamond f(y,x,y) = (y \diamond Sy) \diamond ((x \diamond y) \diamond Sy) = f(x \diamond y, y, Sy)$$

and hence by Equation (10) again

$$(x \diamond y) \diamond y = x \diamond y$$

giving (E378). □

Let E be a law of the form Equation (10) that implies (E378). We define a directed graph $\to_E$ on words in $\mathcal{M}_X$ by declaring $w' \to_E w$ if $w \sim_E w'' \diamond w'$ for some $w' \in \mathcal{M}_X$. By (E378) (applied to the quotient magma $\mathcal{M}_{X,E} = \mathcal{M}_X / \sim_E$), this is equivalent to requiring that $w \sim_E w \diamond w'$. In particular, from Equation (10) we have $f(x,y,z) \to x$ for all x, y, z. Furthermore, the relation $\to_E$ factors through $\sim_E$: if $w \sim_E \tilde{w}$ and $w' \sim_E \tilde{w}'$, then $w' \to_E w$ if and only if $\tilde{w} \to_E \tilde{w}$.

Call a word $w \in M_X$ *irreducible* if it is not of the form $w = w_1 \diamond w_2$ with $w_2 \to_E w_1$. We can partially understand the equivalence relation $\sim_E$ on irreducible words:

Theorem 6.7 (Description of equivalence)**.** *Let E be an equation of the form Equation* (10)*. Let w be an irreducible word, and let w' be a word with $w \sim_E w'$.*

(i) If w is a product $w = w_1 \diamond w_2$, then w' takes the form

$$w' = (((w_1' \diamond w_2') \diamond v_1) \diamond \ldots \diamond v_n)$$

for some $w_1' \sim_E w_1$, $w_2' \sim_E w_2$, some $n \geq 0$, and some words $v_1, \ldots, v_n$ such that for all $0 \leq i < n$, v_{i+1} is of the form

$$v_{i+1} \sim_E f(x_i, y_i, z_i)$$

for some x_i, y_i, z_i with

$$x_i \sim_E (((w_1' \diamond w_2') \diamond v_1) \diamond \ldots \diamond v_i).$$

In particular, $v_{i+1} \to_E x_i$.

(ii) Similarly, if $w \in X$ is a generator of M_X, then w' takes the form

$$w' = ((w \diamond v_1) \diamond \ldots \diamond v_n)$$

for some $n \geq 0$, and some words $v_1, \ldots, v_n$ such that for all $0 \leq i < n$, v_{i+1} is of the form

$$v_{i+1} \sim_E f(x_i, y_i, z_i)$$

for some x_i, y_i, z_i with

$$x_i \sim_E ((w \diamond v_1) \diamond \ldots \diamond v_i).$$

In particular, $v_{i+1} \to_E x_i$.

Conversely, any word of the above forms is equivalent to w.

Proof. We just verify claim (i), as claim (ii) is similar. The converse direction is clear from Equation (10) (after quotienting by $\sim_E$), so it suffices to prove the forward claim. By the Birkhoff completeness theorem, it suffices to prove that the class of words described by (i) is preserved by any term rewriting operation, in which a term in the word is replaced by an equivalent term using Equation (10). Clearly the term being rewritten is in w_1' or w_2' then the form of the word is preserved, and similarly if the term being rewritten is in one of the v_i. The only remaining case is if we are rewriting a term of the form

$$x_i = (((w_1' \diamond w_2') \diamond v_1) \diamond \dots \diamond v_i).$$

If $i > 0$ we can rewrite this term down to x_{i-1}, and this still preserves the required form (decrementing n by one). If $i = 0$ then we cannot perform such a rewriting because of the irreducibility of $w_1 \diamond w_2$ and hence $w_1' \diamond w_2'$. Finally, we can rewrite x_i to $x_i \diamond v$ where v is of the form

$$v_i = f(x_i, y, z),$$

and after some relabeling we are again of the required form (now incrementing n by one). This covers all possible term rewriting operations, giving the claim. □

Specializing to the case where w, w' are both irreducible, we conclude

Corollary 6.8 (Unique factorization)**.** *Two irreducible words w, w' are equivalent if and only if they are either the same generator of X, or are of the form $w = w_1 \diamond w_2$, $w' = w_1' \diamond w_2'$ with $w_1 \sim_E w_1'$ and $w_2 \sim_E w_2'$.*

As an application of this corollary, we establish

Proposition 6.9 (E854 does not imply E3316)**.** *Equation* (E854) *does not imply* (E3316).

Proof. (Sketch) We work in the free group $\mathcal{M}_X$ on two generators $X = \{\mathrm{x}, \mathrm{y}\}$. It suffices to show that

$$\mathrm{x} \diamond \mathrm{y} \not\sim_{E854} \mathrm{x} \diamond (\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{y})).$$

Suppose this were not the case, then by Corollary 6.8 one of the following statements must hold:

(i) $y \to_{E854} x$.
(ii) $(y \diamond (x \diamond y)) \to_{E854} x$.
(iii) $y \diamond (x \diamond y) \sim_{E854} y$.

If (i) holds, then we have $x \diamond y = x$ must hold in $\mathcal{M}_X / \sim_E$, hence (E854) would imply (E4). However, it is possible to refute this implication by a finite counterexample.

Similarly, if (iii) held, then (E854) would have to imply (E10), but this can also be refuted by a finite magma.

Finally, if (ii) held, then the claim

$$x \diamond y \sim x \diamond (y \diamond (x \diamond y))$$

to refute simplifies to

$$x \diamond y \sim x$$

and we are back to (i), which we already know not to be the case. □

7. Proof Automation

In this project we used proof automation in two ways: automated theorem provers (ATPs) and Lean tactics. ATPs are generally stand-alone tools that implement a (semi-) decision procedure for a given formal language or related set of languages. For example, Vampire [22] is an ATP focused primarily on first-order logic using superposition, which we used extensively in this project.

ATPs are complex software that can contain bugs. Instead of trusting ATP output, we used proof certificates, which many ATPs can produce, to reconstruct a proof in Lean. This process depends on the proof certificate and the ATP, and we describe it for the main reconstruction we have done.

Tactics in Lean, on the other hand, are meta-programs [9] that builds proofs. In other words, tactics are programs that operate at the meta-level of Lean code: they essentially take in Lean code as input and produce Lean code as output. In this manner, they look like another keyword in the language, and are tightly integrated by producing proofs directly. Under the hood, they can implement essentially anything, from syntactic sugar to full decision procedures. The `duper` tactic [6], for example, implements a superposition calculus, similar to Vampire's, but for dependent types — Lean's underlying logical foundation.

In the rest of this section we describe the different proof automation techniques and ATPs/tactics we used. We first discuss the different proof methods used: primarily superposition and equational reasoning, we then discuss the integration in Lean, and finally we report some basic empirical results from this project.

7.1. **Proof Techniques.** The main two families of ATPs and tactics we used here are superposition/saturation-based and equational reasoning ones. In this context we also include SMT solvers, which combine specific decision procedures for theories, like congruence closure for equational reasoning, with satisfiability (SAT) solving **Give citation**. Finally, we also used `aesop` [26], which implements a version of tableau search. This was used mainly to help specific constructions in refutations, and is not specific to proving or disproving magma implications in this sense. We will describe our use of `aesop` more in Section 7.2 below.

Saturation. Most of the ATPs we used extensively in this project rely primarily on saturation procedures in the superposition calculus. This is the case for Vampire [22]. See also [4] for a gentler exposition. The core idea of these provers is that they take a set of assumptions with a conjecture, expressed in — say — first-order logic. They take the conjecture and negate it, adding this negation to the set of assumptions, which are all put in some normal form. The ATP then tries to refute the negation by applying rules of the underlying calculus, until they find a proof of false (a contradiction). In this case, the conjecture was (classically) true, and the ATP has found a proof by contradiction, often called a "refutation" or "saturation" proof.

The underlying calculus varies from system to system, but they often have a variant of a resolution clause, a clause of a form:

$$\frac{C \vee L \quad D \vee \neg L}{C \vee D}$$

This can also be read as $C \vee L \quad D \vee \neg L$ implies $C \vee D$, where C, D, L are formulas in e.g. first-order logic. Superposition calculi have a variant of this rule that deals with equality directly, and thus are more efficient at reasoning about equality.

In this project we used Vampire [22], Duper [6] and Prover9 and Mace4 [27] which are all based on variants of saturation for proving. TODO: here we could add a screenshot of using vampire or prover9.

Equational Reasoning. Equational reasoning is a type of reasoning based on equational logic and rewriting with congruence [3], see Chapter ??? for a discussion of its foundations in universal algebra. In general, it takes a series of equations and determines whether another equation can be deduced from it. A core tool in equational reasoning are e-graphs, a data structure used to represent equivalence classes of terms. The procedure of congruence closure that can be used to decide ground equational problems (i.e. problems without variables) and as a semi-decision procedure in general.

SMT solvers like Z3 [8] use equational reasoning for deciding the theory of equality with uninterpreted functions [23, 7]. On the other hand, equality saturation [39] uses e-graphs by extending congruence closure to a more controlled search, enabling optimization and conditional rewriting. One of the main advantages of using equational reasoning to reason about implications of magma laws is that we get very explicit proofs: a proof that $l \vdash l'$ is given by a sequence of rewrites that starts at the left-hand side of l' and arrives at the right-hand side through applications of l.

In this project we used Z3 [8], Prover9 and Mace4 [27], a custom ATP for magmas based on egg [39], and the Lean egg tactic [20, 35], which all work with equational logic. We have also reasoned with manual (custom written) heuristics about simple rewrites.

7.2. **Proof Reconstruction and Integration.** While ATPs are very useful to solve questions for this project, they generally don't integrate well with Lean . A bug in an ATP could lead to an unsound proof, or worse, an incorrect result. To avoid having to trust ATP's large codebases, we take results found by the different provers and (re-)construct their proofs in Lean.

An exception for this are tactics like `duper`, `aesop` and `egg`. Figure 4 shows an example of the `egg` tactic as used in this project. It integrates directly in Lean, generating a Lean proof directly. With the variant `egg?`, depicted in the screenshot, it uses an auxiliary tactic

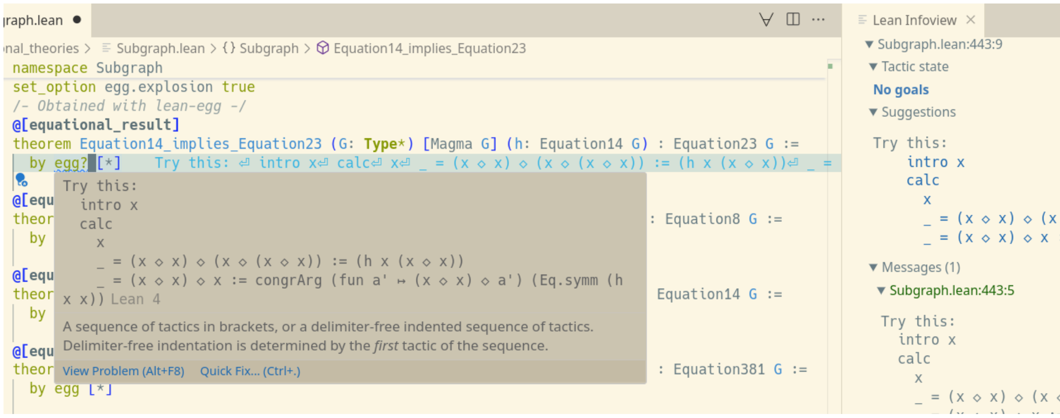

FIGURE 4. An example of the use of proof automation with tactics. This shows the egg tactic as it was used to generate (human-readable) equational proofs of positive implications.

`calcify`[6] to generate a human-readable proof as a series of calculation steps, which can be incorporated into the file with a single click.

In general, integration implies two steps: invoking the decision procedures/ATPs (translating the problem from Lean into the languages and logics they use), and conversely, (re-)constructing the results from the decision procedure as a (persistent) Lean proof. These two aspects present different challenges, and require different strategies, depending mostly on the kind of proof strategy the decision procedure uses.

For saturation proofs ... (TODO: add explanations from `https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Vampire.20-.3E.20Lean/near/478147138`)

For equational proofs from external provers (e.g. MagmaEgg), we used also a simple version of reconstruction by (re-)constructing the proofs of equality from an explanation, using congruence lemmas from Lean. In the case of equational proofs from the `egg` tactic, these could be converted into a series of `calc` steps, documenting the explicit calculation, using the `calcify` tactic[7].

In general, we have observed that there are multiple ways of integrating decisions procedures within Lean, with different levels of integration.

(1) Using a Lean tactic, which calls a decision procedure written in Lean (like `aesop` or `duper`).
(2) Using a Lean tactic, which calls an existing (external) ATP and reconstructs a proof term from the ATP's result (like `bv_decide` or `egg`).
(3) Using an external script which calls an existing ATP and generates a source file `.lean` which captures the result explicitly.

[6] `https://github.com/nomeata/lean-calcify`
[7] `https://github.com/nomeata/lean-calcify`

This project primarily used the least integrated approach, (Option 3), as it was fastest and required no dependencies on the other contributors. This also has drawbacks: primarily, while the upfront effort is lower, the effort to use is higher than with a tactic, once the tactic is developed. It also makes integration with larger code bases more difficult: in this project the (mathematical) dependencies were by design very minimal, magmas are simple, and we built our own definitions for them. For example, integration with the typeclass system becomes much more difficult when working with more complex mathematical objects that build on multiple, nested layers of structure in non-trivial ways. In that case, for example, tactics need to synthesize typeclass instances, deal with diamonds and different notions of equivalence [38].

Semi-Automated Counterexample Guidance. Another use of ATPs has been in a semi-automatic fashion, to find counter-examples. The general strategy was to use ATPs to find counter-examples to implications by building magmas iteratively. If we want to build a counterexample to $l \vdash l'$, we want to construct a magma where l holds but l' does not. In this method, we iteratively strengthen a construction with additional hypotheses, and use the ATP to check whether these hypotheses are not too strong (to imply l') or unsound (to disallow l).

TODO: this should also be expanded more, at least with references to some of the constructions in other chapters.

While equational reasoning can also be used in a semi-automatic fashion to prove equations [20], the positive implications in the main implication graph of project were all simple enough that we did not need a semi-automatic approach for them. TODO: discuss guided search in the finite implications or the Higman-Neumann work Jose Brox has done.

TODO: maybe add a screenshot here of the workflow of using a seed to find counterexamples with prover9 or vampire?

7.3. **Empirical Results.** Finally, we report some empirical results from use of ATPs for this project, in terms of performance. The aim of this section is not to be a careful evaluation and benchmark comparison of the different ATPs; instead, we present our work here as a more informal “field report” documenting our experiences. In particular, we don’t believe we can draw firm conclusions about the overall capabilities of the different ATPs. Rather, this serves as a use-case documenting the experience of (mostly) novice users.

TODO: throw a couple of ”benchmarking” tables for the same ATP with different parameters and for different ATPs, talk about some relative gains in time (changing parameters we saw a 500 times speedup on this particular problem), etc. This is knowledge I think we have gained to some extent, and certainly I would have been glad to receive this kind of hints before we started!”. Then leave it as an interesting open problem to properly develop and measure benchmarks for ATPs based on this project.

TODO: Any comparative study of semi-automated methods with fully automated ones? In principle, the semi-automated approach could be more automated using a script or ”agent” to call various theorem provers. See this discussion.

Note: when evaluating the performance of any particular automated implication tool, we should do a fresh run on the entire base of implications, rather than rely on whatever implications produced by the tool remain in the Lean codebase by the time of writing the paper. This is because (a) many of the previous runs focused only on those implications that were not already ruled out by earlier contributions, and (b) some pruning has been applied subsequent to the initial runs to improve compilation performance. Of course, these runs would not need to be added to the (presumably optimized) codebase at that point, but would be purely for the purpose of gathering performance statistics. More discussion here.

See this discussion on the value of using different ATPs and setting run time parameters etc. at different values.

What are the hardest implications to prove? See this discussion.

Make a note of the possible alternate strategy to prove implications outlined here.

8. Implications for Finite Magmas

TODO: Expand this sketch

Introduce Austin pairs.

Recap discussion from `https://leanprover.zulipchat.com/#narrow/channel/458659-Equational/topic/Austin.20pairs`

9. Higman–Neumann laws

TODO: report on Higman–Neumann laws

10. AI and Machine Learning Contributions

TODO: expand this sketch

- Claude assistance in coding front ends.
- ChatGPT to guess a complete rewriting system.
- Minor use of GitHub Copilot to autocomplete code in Lean and other languages.
- See this discussion.
- Directed Link Prediction (DLP) on the implication graph with multiple GNN autoencoders (see related Zulip topic).

ML experiments to learn the implication graph.

10.1. **Graph ML: Directed link prediction on the implication graph.** We experimented with various Graph Neural Network (GNN) autoencoders to predict missing edges, providing a way to estimate the truth value of unproven implications. To assess these models, we defined three test sets focusing on edge existence, directionality, and bidirectionality. The results give insight into how these models handle dense, directed graphs like ours. A more detailed report follows below.

10.1.1. *Motivation.* Directed Link Prediction [14] is a method enabling machine learning and deep learning models to predict missing edges in a directed graph. For our implication graph, this translates to predicting the truth values of unproven implications. This task serves as a necessary first step for advancing in the following directions:

(1) **Reasoning over mathematical knowledge graphs:** Recent advancements allow language models to integrate information from multiple modalities. For example, [42, 41] share information between corresponding layers of Language Models (LMs) and Graph Neural Networks (GNNs), enabling simultaneous learning from text corpora and graph-structured data within the same expert domain. By leveraging both modalities, the language model can better *structure* its knowledge and respond to complex queries. This dual learning process combines masked language modeling for text with link prediction on the graph, highlighting the importance of link prediction for robust reasoning.

(2) **Higher-order implication graphs:** Our implication graph currently represents only implications of the form $p \implies q$, not more complex ones like $(p \wedge r) \implies q$. Extending to such higher-order edges would likely involve connecting sets of nodes, thereby requiring hypergraph representations. For a systematic overview, see [17]. While specific hypergraph neural architectures exist [11], we believe it is still conceptually important to apply Directed Link Prediction to simpler implication graphs first, providing insights and guiding principles that can anticipate challenges in higher-order graph representation learning.

10.1.2. *Data.* We used this edge list, generated on October 20, 2024, with the following commands:

```
lake exe extract_implications outcomes > data/tmp/outcomes.json
scripts/generate_edgelist_csv.py
```

The structure of the implication graph is summarized below:

```
graph_summary = {
    "total_nodes": 4694,
    "total_directed_edges": 8178279,
    "edge_density_percentage": 37,  # Percentage of possible edges that exist
    "bidirectional_edges": 2475610,
    "bidirectional_percentage": 30  # Percentage of all edges that are bidirectional
}
```

Below is a summary of the edge types in the graph:

```
edge_counts = {
    "explicit_conjecture_false": 92,
    "explicit_proof_false": 582316,
    "explicit_proof_true": 10657,
    "implicit_conjecture_false": 142,
    "implicit_proof_false": 13272681,
    "implicit_proof_true": 8167622,
    "unknown": 126
}
```

Edges are labeled according to the following scheme:

```
edge_labels = {
    "implicit_proof_true": 1,
    "explicit_proof_false": 0,
    "implicit_proof_false": 0,
    "explicit_proof_true": 1,
    "explicit_conjecture_false": 0,
    "implicit_conjecture_false": 0,
    "unknown": 0
}
```

The `unknown` class contains a very small number of edges (126), so their impact on the training phase is expected to be negligible. Future approaches might address this class by excluding `unknown` edges from the training set.

10.1.3. *Methods.* Consider a directed graph $G = (V, E)$ where $E = \{(u, v) \mid u, v \in V\}$ is the edge set and $|V| = n$. We assume that each node is associated with a feature vector, resulting in an $X \in \mathbb{R}^{n \times f}$ feature matrix.

We define the existing edges $(a, b) \in E$ as *positives* and the non-existing edges $(c, d) \notin E$ as *negatives*.

Intuitively, performing Directed Link Prediction (DLP) on G involves randomly splitting E into three disjoint sets: E_{train}, E_{val}, and E_{test}, such that:

- E_{train} is the training set,
- E_{val} is the validation set,
- E_{test} is the test set,
- and $E = E_{\text{train}} \dot\cup E_{\text{val}} \dot\cup E_{\text{test}}$.

The model then learns from $G_{\text{train}} = (V, E_{\text{train}})$ to map the topological and feature-related information of two nodes u and v to a probability p_{uv} that $(u, v) \in E_{\text{test}}$.

However, this setup presents two key issues among others:

(1) The model learns only from *positives*, so it cannot recognize *negatives*.
(2) The model is evaluated only on *positives*, preventing us from measuring its ability to identify *negatives*.

To address these limitations, we adopted the setup proposed in [36]. Specifically, we redefined E_{train} to E^p_{train} (*positives*) and introduced:

$$E_{\text{train}} = E^p_{\text{train}} \dot{\cup} E^n_{\text{train}}$$

where E^n_{train} includes all possible *negatives* in $G_{\text{train}} = (V, E^p_{\text{train}})$. The model is now required to predict the non-existence of edges in E^n_{train}.

Similarly, if we redefine E_{test} as follows:

$$E_{\text{test}} = E^p_{\text{test}} \dot{\cup} E^n_{\text{test}}$$

where E^n_{test} is a *random* sample of *negatives*, the model's evaluation would fail to capture two crucial aspects:

(1) The model's ability to distinguish (u, v) from (v, u) for all $(u, v) \in E^p_{\text{test}}$.
(2) The model's ability to identify bi-implications.

These limitations arise from the random selection of negative edges in E^n_{test}. To address this, we define three distinct test sets: E^G_{test}, E^D_{test}, and E^B_{test}, to evaluate different facets of the model's performance:

- **General Test Set** (E^G_{test}): Here, $E_{\text{test}} = E^p_{\text{test}} \dot{\cup} E^n_{\text{test}}$, where E^n_{test} is a random sample of non-existent edges with the same cardinality as E^p_{test}. This set assesses the model's ability to detect the presence of edges, regardless of direction. A model that cannot distinguish edge direction may still perform well on this set, highlighting the need for the following two additional test sets.
- **Directional Test Set** (E^D_{test}): Defined as $E^{\text{up}}_{\text{test}} \dot{\cup} \tilde{E}^{\text{up}}_{\text{test}}$, where $E^{\text{up}}_{\text{test}}$ consists of unidirectional edges in E^p_{test}, and $\tilde{E}^{\text{up}}_{\text{test}}$ contains their reverses (negatives by construction). This set evaluates the model's ability to recognize edge direction, making it suitable for assessing direction-sensitive models.
- **Bidirectional Test Set** (E^B_{test}): Defined as $E^{\text{bp}}_{\text{test}} \dot{\cup} E^{\text{n}}_{\text{B}}$, where $E^{\text{bp}}_{\text{test}}$ contains one direction of all bidirectional edges in E^p_{test}, and $E^{\text{n}}_{\text{B}} \subset \tilde{E}$ includes a subset of their reverses. This set evaluates the model's ability to identify bi-implications, as each edge in E^B_{test} has a reverse that is positive, but only half are bidirectional in practice.

We tested the following models:

- **GAE** [14]
- **Gravity-GAE** [36]
- **Source/Target-GAE** [36]
- **DiGAE** [21]
- **MagNet** [43]

All these models are graph-based autoencoders with distinct encoder-decoder architectures. Notably, GAE is the only model structurally unable to differentiate edge directions. Each model outputs the probability that an ordered pair of nodes has a directed edge between them, with nodes represented using one-hot encodings as features.

We trained the models using Binary Cross Entropy as the loss function, balancing the contribution of positive and negative edges through re-weighting. On the *General* test set, we evaluated the following metrics:

- **AUC** (Area Under the ROC Curve): Measures the probability that the model ranks a random positive edge higher than a random negative edge. Higher values indicate better discrimination between positive and negative edges.
- **AUPRC** (Area Under Precision-Recall Curve): Assesses model performance, particularly in cases of class imbalance. AUPRC is more robust to imbalanced data than AUC.
- **Hits@K**: Evaluates the fraction of times a positive edge is ranked within the top K positions among personalized negative samples [25]. Briefly, given a positive edge, its M personalized negative samples are M negative edges with the same head but different tails. We calculate Hits@K for $K = 1, 3, 10$ to assess ranking quality, and set $M = 100$. Therefore, Hits@K = 0.1 means that on average, the model ranks a positive edge within the highest-ranked K personalized negatives 10% of the time.
- **MRR** (Mean Reciprocal Rank): Computes the average reciprocal rank of positive edges among their personalized negative samples [25] (the same as those used for Hits@K) providing an overall measure of ranking accuracy. For instance, $MRR = 0.1$ means that on average, the model ranks a positive edge as 10^{th} among M personalized negative samples.

Each metric ranges from 0 to 1, with higher values reflecting improved performance. Based on prior work, we expect AUC and AUPRC scores to approach 1, while MRR and Hits@K often yield values around 0.15 for similar undirected tasks [25]. *Directional* and *Bidirectional* performances were evaluated using only AUC and AUPRC, since Hits@K and MRR are hardly definable in those scenarios. The number of training epochs was optimized through Early Stopping on the *General* validation set performance (given by the sum of AUC and AUPRC).

10.1.4. *Results.* The results below represent average performance over six random splits of E_{train}, E_{val}, and E_{test} while keeping the model's seed fixed for fair comparison. The *training / validation / test* split proportions are:

- 85/5/10 for unidirectional edges,

- 65/15/30 for bidirectional edges.

Model	G_ROC_AUC	G_AUPRC	G_Hits@1	G_Hits@3
`gae`	$0.8484 \pm 9 \times 10^{-4}$	$0.8558 \pm 6 \times 10^{-4}$	$6 \times 10^{-5} \pm 4 \times 10^{-5}$	$6 \times 10^{-5} \pm 4 \times 10^{-5}$
`gravity_gae`	$0.9806 \pm 3 \times 10^{-4}$	$0.9753 \pm 4 \times 10^{-4}$	$0.069 \pm 6 \times 10^{-3}$	$0.101 \pm 5 \times 10^{-3}$
`sourcetarget_gae`	$0.99976 \pm 1 \times 10^{-5}$	$0.999736 \pm 8 \times 10^{-6}$	$0.077 \pm 4 \times 10^{-3}$	$0.147 \pm 7 \times 10^{-3}$
`mlp_gae`	$0.99315 \pm 1 \times 10^{-5}$	$0.99409 \pm 1 \times 10^{-5}$	$0.181 \pm 7 \times 10^{-3}$	$0.299 \pm 7 \times 10^{-3}$
`digae`	$0.9978 \pm 3 \times 10^{-4}$	$0.998 \pm 3 \times 10^{-4}$	$0.035 \pm 6 \times 10^{-3}$	$0.068 \pm 1 \times 10^{-2}$
`magnet`	$0.989 \pm 1 \times 10^{-4}$	$0.99076 \pm 3 \times 10^{-5}$	$0.151 \pm 1 \times 10^{-2}$	$0.26 \pm 2 \times 10^{-2}$

TABLE 2. Results for various graph autoencoder models.

10.1.5. *Discussion.* We observe consistently high *General* AUC and AUPRC scores, close to 1. These high values are expected, as similar neural architectures have demonstrated strong performance in graphs of comparable size [14]. The high ratio of existing to non-existing edges in the implication graph (approximately 37%) likely contributes to the near-perfect *General* AUC and AUPRC scores. For context, benchmark datasets such as Cora and Citeseer (e.g., directed and undirected) contain fewer than 1% of all possible edges.

Interestingly, the GAE model, though structurally unable to distinguish edge direction, performs well on the *General* task (if we consider AUC and AUPRC only). This experimentally confirms the need for including *Directional* and *Bidirectional* test sets, which allow comprehensive evaluation across all facets of Directed Link Prediction (DLP).

All other models demonstrate high AUC and AUPRC scores across the *General*, *Directional*, and *Bidirectional* test sets, indicating strong predictive capabilities even when accounting for directionality and bidirectionality.

Notably, the `mlp_gae` and `magnet` models also achieve competitive scores in MRR and Hits@K metrics.

10.1.6. *Conclusions.* We evaluated the performance of six different graph autoencoders on a Directed Link Prediction (DLP) task. By adopting a multi-task evaluation framework, we assessed the models comprehensively across various aspects of DLP. All models demonstrated strong performance on AUC and AUPRC metrics, with some also achieving high scores on MRR and Hits@K.

Node features were represented using one-hot encodings, meaning that the models received no explicit information about the equations represented by the nodes. Instead, they relied entirely on the topological structure encoded during training. This approach aligns with previous research suggesting that one-hot encodings can promote asymmetric embeddings [36]. However, future experiments could explore alternative representations, such as encoding the equations with text-based embeddings like Word2Vec, to potentially enhance the models' understanding of the nodes' semantic content.

In summary, our findings highlight the adaptability and robustness of graph autoencoders for DLP tasks in dense, directed graphs. This approach not only demonstrates robustness in predicting directed links but also suggests promising potential for future improvements through enhanced feature representations, thereby advancing the capabilities of link prediction in complex mathematical knowledge graphs.

11. User Interface

Describe visualizations and explorer tools: Equation Explorer, Finite Magma Explorer, Graphiti, ...

12. Statistics and Experiments

TODO: Expand this sketch

Some statistics are discussed here.

Data analysis of the implication graph:

- Mention the long chain $2 \Rightarrow 5 \Rightarrow 2499 \Rightarrow 2415 \Rightarrow 238 \Rightarrow 2716 \Rightarrow 28 \Rightarrow 2973 \Rightarrow 270 \Rightarrow 3 \Rightarrow 3715 \Rightarrow 375 \Rightarrow 359 \Rightarrow 4065 \Rightarrow 1$ (discussed here).
- What are the "most difficult" implications?
- Is there a way to generate a standard test set of implication problems of various difficulty levels? Can one then use this to benchmark various automated and semi-automated methods? Challenge: how does one automatically assign a difficulty level to a given (anti-)implication?

See this for a preliminary data analysis of the impact of equation size and the number of variables.

Analyze the implication graph and discuss test sets of implication problems for benchmarking theorem provers. Challenge: How can one automatically assign a difficulty level to an implication?

13. Data Management

TODO: expand this sketch

Describe how data was handled during the project and how it will be managed going forward.

14. Conclusions and Future Directions

TODO: Expand this sketch

Insights learned, future speculation. Utilize the discussion on future directions. Some ideas from that list:

- A database of triple implications (EquationX and EquationY imply EquationZ) - see also this discussion.
- Are there any equational laws that have no non-trivial finite models, but have surjunctive models?
- Can we produce interesting examples of irreducible implications EquationX -¿ EquationY (no intermediate EquationZ can interpose)
- Degree of satisfiability results, e.g., if a central groupoid obeys the natural central groupoid axiom 99% of the time, is it a natural central groupoid?
- Kisielewicz's question: does 5093 have any infinite models?
- "Insight mining" the large corpus of automated proofs that have been generated.
- How machine-learnable is the implication graph? (See AI/ML section)

Note that our choice to focus particularly on some laws and not others is to some extent an artefact of the order in which we discovered and deployed tools. For instance, by deploying automated theorem provers at an early stage, we might have settled some implications that had more interesting human-readable proofs that we missed. Similarly, we developed some sophisticated theory for 854 that is now superseded by finite counterexamples; but had the finite counterexamples been discovered first, we would not have found the theoretical arguments. It may be productive for future work to revisit some portions of the implication graph and locate alternate proofs and methods.

Automation often overtook the rate of human progress, for instance in developing metatheorems. Does this create an opportunity cost? Raised as a possible discussion point here by Zoltan Kocsis.

Acknowledgments

Thanks to Claudio Moroni for the exploration of directed link prediction on the implication graph using GNN autoencoders described in Section 10.1.

We are also grateful to the many additional participants to the Equational Theories Project for their numerous comments and encouragement, including but not restricted to Edward van de Meent, ...

Appendix A. Numbering system

In this section we record the numbering conventions we use for equational laws.

For this formal definition we use the natural numbers $0, 1, 2, \dots$ to represent and order indeterminate variables; however, in the main text, we use the symbols $x, y, z, w, u, v, r, s, t$ instead (and do not consider any laws with more than eight variables).

We extend the ordering on indeterminate variables to a well-ordering on words w in the free magma generated by these variables by declaring $w > w'$ if one of the following holds:

- w has a larger order than w'.
- $w = w_1 \diamond w_2$ and $w' = w'_1 \diamond w'_2$ have the same order $n \geq 1$ with $w_1 > w'_1$.
- $w = w_1 \diamond w_2$ and $w' = w'_1 \diamond w'_2$ have the same order $n \geq 1$ with $w_1 = w'_1$ and $w_2 > w'_2$.

Thus, for instance

$$0 < 1 < 0 \diamond 0 < 0 \diamond 1 < 1 \diamond 0$$

and

$$1 \diamond 1 < 0 \diamond (0 \diamond 0) < (0 \diamond 0) \diamond 0.$$

We similarly place a well-ordering on equational laws $w_1 \simeq w_2$ by declaring $w_1 \simeq w_2 > w'_1 \simeq w'_2$ if one of the following holds: as follows:

- $w_1 \simeq w_2$ has a longer order than $w'_1 \simeq w'_2$.
- If $w_1 \simeq w_2$ has the same order as $w'_1 \simeq w'_2$, and $w_1 > w'_1$.
- If $w_1 \simeq w_2$ has the same order as $w'_1 \simeq w'_2$, $w_1 = w'_1$, and $w_2 > w'_2$.

Two equational laws are equivalent if one can be obtained from another by some combination of relabeling the variables and applying the symmetric law $w_1 \simeq w_2 \iff w_2 \simeq w_1$. For instance, $(0 \diamond 1) \diamond 2 \simeq 1$ is equivalent to $0 \simeq (1 \diamond 0) \diamond 2$. We then replace every equational law with their minimal element in their equivalence class, which can be viewed as the *normal form* for that law; for instance, the normal form of $(0 \diamond 1) \diamond 2 \simeq 1$ would be $0 \simeq (1 \diamond 0) \diamond 2$. Finally, we eliminate any law of the form $w \simeq w$ other than $0 \simeq 0$. We then number the remaining equations $E1, E2, \ldots$. For instance, $E1$ is the trivial law $0 \simeq 0$, $E2$ is the constant law $0 \simeq 1$, $E3$ is the idempotent law $0 \simeq 0 \diamond 0$, and so forth. Lists and code for generating these equations, or the equation number attached to a given equation, can be found in the ETP repository.

The number of equations in this list of order $n = 0, 1, 2, \ldots$ is given by

$$2, 5, 39, 364, 4284, 57882, 888365, \ldots$$

(`https://oeis.org/A376640`). The number can be computed to be

$$C_{n+1}B_{n+2}/2$$

if n is odd, 2 if $n = 0$, and

$$(C_{n+1}B_{n+2} + C_{n/2}(2D_{n+2} - B_{n+2}))/2 - C_{n/2}B_{n/2+1}$$

if $n > 2$ is even, where C_n, B_n are the Catalan and Bell numbers, and D_n is the number of partitions of $[n]$ up to reflection, which for $n = 0, 1, 2, \ldots$ is

$$1, 1, 2, 4, 11, 32, 117, \ldots$$

(`https://oeis.org/A103293`). A proof of this claim can be found in the ETP blueprint. In particular, there are 4694 equations of order at most 4.

Below we record some specific equations appearing in this paper, using the alphabet $\mathrm{x}, \mathrm{y}, \mathrm{z}, \mathrm{w}, \mathrm{u}, \mathrm{v}$ in place of $0, 1, 2, 3, 4, 5, \ldots$ for readability.

(E1)	$\mathrm{x} \simeq \mathrm{x}$	(Trivial law)
(E2)	$\mathrm{x} \simeq \mathrm{y}$	(Singleton law)
(E3)	$\mathrm{x} \simeq \mathrm{x} \diamond \mathrm{x}$	(Idempotent law)
(E4)	$\mathrm{x} \simeq \mathrm{x} \diamond \mathrm{y}$	(Left-absorptive law)
(E5)	$\mathrm{x} \simeq \mathrm{y} \diamond \mathrm{x}$	(Right-absorptive law)
(E10)	$\mathrm{x} \simeq \mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{x})$	
(E23)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond \mathrm{x}$	
(E41)	$\mathrm{x} \diamond \mathrm{x} \simeq \mathrm{y} \diamond \mathrm{z}$	
(E43)	$\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{y} \diamond \mathrm{x}$	(Commutative law)
(E46)	$\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{z} \diamond \mathrm{w}$	(Constant law)
(E73)	$\mathrm{x} \simeq \mathrm{y} \diamond (\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{y}))$	
(E151)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond \mathrm{x})$	
(E168)	$\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond \mathrm{z})$	(Central groupoid law)
(E206)	$\mathrm{x} \simeq (\mathrm{x} \diamond (\mathrm{x} \diamond \mathrm{y})) \diamond \mathrm{y}$	
(E327)	$\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z})$	
(E378)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{y}$	
(E395)	$\mathrm{x} \diamond \mathrm{y} \simeq (\mathrm{z} \diamond \mathrm{x}) \diamond \mathrm{y}$	
(E543)	$\mathrm{x} \simeq \mathrm{y} \diamond ((\mathrm{z} \diamond (\mathrm{x} \diamond (\mathrm{y} \diamond \mathrm{z}))))$	(Tarski's axiom)
(E854)	$\mathrm{x} \simeq \mathrm{x} \diamond ((\mathrm{y} \diamond \mathrm{z}) \diamond (\mathrm{x} \diamond \mathrm{z}))$	
(E1110)	$\mathrm{x} \simeq \mathrm{y} \diamond ((\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{x})) \diamond \mathrm{y})$	
(E1117)	$\mathrm{x} \simeq \mathrm{y} \diamond ((\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{z})) \diamond \mathrm{z})$	
(E1286)	$\mathrm{x} \simeq \mathrm{y} \diamond (((\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{x}) \diamond \mathrm{y})$	
(E1485)	$\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond (\mathrm{x} \diamond (\mathrm{z} \diamond \mathrm{y}))$	(Weak central groupoids)
(E1571)	$\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{z}) \diamond (\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{z}))$	(Exp. 2 abelian groups)
(E1629)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{x}) \diamond ((\mathrm{x} \diamond \mathrm{x}) \diamond \mathrm{x})$	
(E1648)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{y}) \diamond ((\mathrm{x} \diamond \mathrm{y}) \diamond \mathrm{y})$	
(E1659)	$\mathrm{x} \simeq (\mathrm{x} \diamond \mathrm{y}) \diamond ((\mathrm{y} \diamond \mathrm{y}) \diamond \mathrm{z})$	
(E1689)	$\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{x}) \diamond ((\mathrm{x} \diamond \mathrm{z}) \diamond \mathrm{z})$	
(E1729)	$\mathrm{x} \simeq (\mathrm{y} \diamond \mathrm{y}) \diamond ((\mathrm{y} \diamond \mathrm{x}) \diamond \mathrm{y})$	
(E2301)	$\mathrm{x} \simeq (\mathrm{y} \diamond (\mathrm{x} \diamond (\mathrm{y} \diamond x))) \diamond y$	
(E2441)	$\mathrm{x} \simeq (\mathrm{x} \diamond ((\mathrm{x} \diamond \mathrm{x}) \diamond \mathrm{x})) \diamond \mathrm{x}$	
(E3316)	$\mathrm{x} \diamond \mathrm{y} \simeq \mathrm{x} \diamond (\mathrm{y} \diamond (\mathrm{x} \diamond \mathrm{y}))$	

(E4315) $x \diamond (y \diamond x) \simeq x \diamond (y \diamond z)$

(E4380) $x \diamond (x \diamond x) \simeq (x \diamond x) \diamond x$

(E4482) $x \diamond (y \diamond y) = (y \diamond y) \diamond x$

(E4512) $(x \diamond y) \diamond z \simeq x \diamond (y \diamond z)$ (Associative law)

(E4531) $x \diamond (y \diamond z) \simeq (y \diamond z) \diamond x$

(E345169) $x \simeq (y \diamond ((x \diamond y) \diamond y)) \diamond (x \diamond (z \diamond y))$ (Sheffer stroke)

(E42323216) $$x \simeq y \diamond ((((y \diamond y) \diamond x) \diamond z) \diamond (((y \diamond y) \diamond y) \diamond z))$$ (Division in groups)

Appendix B. Author Contributions

In this section we list the authors of this paper, grant support, and their contributions to this project, using the following standard CRediT categories:

(1) Conceptualization: Ideas; formulation or evolution of overarching research goals and aims.
(2) Data Curation: Management activities to annotate (produce metadata), scrub data and maintain research data (including software code, where it is necessary for interpreting the data itself) for initial use and later reuse.
(3) Formal Analysis: Application of statistical, mathematical, computational, or other formal techniques to analyze or synthesize study data.
(4) Funding Acquisition: Acquisition of the financial support for the project leading to this publication.
(5) Investigation: Conducting a research and investigation process, specifically performing the experiments, or data/evidence collection.
(6) Methodology: Development or design of methodology; creation of models.
(7) Project Administration: Management and coordination responsibility for the research activity planning and execution.
(8) Resources: Provision of study materials, reagents, materials, patients, laboratory samples, animals, instrumentation, computing resources, or other analysis tools.
(9) Software: Programming, software development; designing computer programs; implementation of the computer code and supporting algorithms; testing of existing code components.
(10) Supervision: Oversight and leadership responsibility for the research activity planning and execution, including mentorship external to the core team.
(11) Validation: Verification, whether as a part of the activity or separate, of the overall replication/reproducibility of results/experiments and other research outputs.
(12) Visualization: Preparation, creation and/or presentation of the published work, specifically visualization/data presentation.
(13) Writing - Original Draft Preparation: Creation and/or presentation of the published work, specifically writing the initial draft (including substantive translation).

(14) Writing - Review and Editing: Preparation, creation and/or presentation of the published work by those from the original research group, specifically critical review, commentary or revision - including pre- or post-publication stages.

- ...
- Pietro Monticone, Department of Mathematics, University of Trento, pietro.monticone@studenti.uni Data Curation, Formal Analysis, Project Administration, Resources, Software, Validation, Writing - Original Draft Preparation, Writing - Review and Editing.
- Terence Tao, Department of Mathematics, UCLA, tao@math.ucla.edu: Conceptualization, Data Annotation, Investigation, Methodology, Project Administration, Writing - Original Draft Preparation, Writing - Review and Editing. Supported by NSF grant DMS-2347850.
- Harald Husum, harald.husum@gmail.com: Investigation, Software, Visualization
- Jérémy Scanvic, Laboratoire de Physique, École Normale Supérieure de Lyon, jeremy.scanvic@ens-lyon.fr: Validation.
- ...

References

[1] A. K. Austin. A note on models of identities. *Proc. Amer. Math. Soc.*, 16:522–523, 1965.

[2] A. K. Austin. Finite models for laws in two variables. *Proc. Amer. Math. Soc.*, 17:1410–1412, 1966.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, Cambridge, 1998.

[4] Alexander Bentkamp, Jasmin Blanchette, Visa Nummelin, Sophie Tourret, Petar Vukmirovic, and Uwe Waldmann. Mechanical mathematicians. *Commun. ACM*, 66(4):80–90, 2023.

[5] George M Bergman. The diamond lemma for ring theory. *Advances in Mathematics*, 29(2):178–218, 1978.

[6] Joshua Clune, Yicheng Qian, Alexander Bentkamp, and Jeremy Avigad. Duper: A proof-producing superposition theorem prover for dependent type theory. In Yves Bertot, Temur Kutsia, and Michael Norrish, editors, *15th International Conference on Interactive Theorem Proving, ITP 2024, September 9-14, 2024, Tbilisi, Georgia*, volume 309 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[7] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Efficient e-matching for SMT solvers. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2007.

[8] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[9] Gabriel Ebner, Sebastian Ullrich, Jared Roesch, Jeremy Avigad, and Leonardo de Moura. A metaprogramming framework for formal verification. *Proc. ACM Program. Lang.*, 1(ICFP):34:1–34:29, 2017.

[10] Trevor Evans. Products of points—some simple algebras and their identities. *Amer. Math. Monthly*, 74:362–372, 1967.

[11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3558–3565, July 2019.

[12] Timothy Gowers and Michael Nielsen. Massively collaborative mathematics. *Nature*, 461(7266):879–881, October 2009.

[13] Graham Higman and B. H. Neumann. Groups as groupoids with one law. *Publ. Math. Debrecen*, 2:215–221, 1952.
[14] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. 2016.
[15] A. Kisielewicz. Austin identities. *Algebra Universalis*, 38(3):324–328, 1997.
[16] Andrzej Kisielewicz. Varieties of algebras with no nontrivial finite members. In *Lattices, semigroups, and universal algebra (Lisbon, 1988)*, pages 129–136. Plenum, New York, 1990.
[17] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, July 2014.
[18] Donald E. Knuth. Notes on central groupoids. *J. Combinatorial Theory*, 8:376–390, 1970.
[19] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 263–297. Pergamon, Oxford-New York-Toronto, Ont., 1970.
[20] Thomas Koehler, Andrés Goens, Siddharth Bhat, Tobias Grosser, Phil Trinder, and Michel Steuwer. Guided equality saturation. *Proc. ACM Program. Lang.*, 8(POPL):1727–1758, 2024.
[21] Georgios Kollias, Vasileios Kalantzis, Tsuyoshi Idé, Aurélie Lozano, and Naoki Abe. Directed graph auto-encoders. 2022.
[22] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.
[23] Daniel Kroening and Ofer Strichman. *Decision Procedures - An Algorithmic Point of View, Second Edition*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2016.
[24] André Kündgen, Gregor Leander, and Carsten Thomassen. Switchings, extensions, and reductions in central digraphs. *J. Combin. Theory Ser. A*, 118(7):2025–2034, 2011.
[25] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. 2023.
[26] Jannis Limperg and Asta Halkjær From. Aesop: White-box best-first proof search for lean. In Robbert Krebbers, Dmitriy Traytel, Brigitte Pientka, and Steve Zdancewic, editors, *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2023, Boston, MA, USA, January 16-17, 2023*, pages 253–266. ACM, 2023.
[27] W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.
[28] William McCune. Solution of the Robbins problem. *J. Automat. Reason.*, 19(3):263–276, 1997.
[29] William McCune. Single axioms: with and without computers. In *Computer mathematics (Chiang Mai, 2000)*, volume 8 of *Lecture Notes Ser. Comput.*, pages 83–89. World Sci. Publ., River Edge, NJ, 2000.
[30] William McCune, Robert Veroff, Branden Fitelson, Kenneth Harris, Andrew Feist, and Larry Wos. Short single axioms for Boolean algebra. *J. Automat. Reason.*, 29(1):1–16, 2002.
[31] Ralph McKenzie. On spectra, and the negative solution of the decision problem for identities having a finite nontrivial model. *J. Symbolic Logic*, 40:186–196, 1975.
[32] N. S. Mendelsohn and R. Padmanabhan. Minimal identities for Boolean groups. *J. Algebra*, 34:451–457, 1975.
[33] C. A. Meredith and A. N. Prior. Equational logic. *Notre Dame J. Formal Logic*, 9:212–226, 1968.
[34] J. D. Phillips and Petr Vojtˇ echovský. The varieties of loops of Bol-Moufang type. *Algebra Universalis*, 54(3):259–271, 2005.
[35] Marcus Rossel and Andrés Goens. Bridging syntax and semantics of lean expressions in e-graphs. *arXiv preprint arXiv:2405.10188*, 2024.
[36] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. Gravity-inspired graph autoencoders for directed link prediction. 2019.
[37] Alfred Tarski. Ein beitrag zur axiomatik der abelschen gruppen. *Fundamenta Mathematicae*, 30(1):253–256, 1938.
[38] Eric Wieser. Multiple-inheritance hazards in dependently-typed algebraic hierarchies. In Catherine Dubois and Manfred Kerber, editors, *Intelligent Computer Mathematics - 16th International Conference, CICM 2023, Cambridge, UK, September 5-8, 2023, Proceedings*, volume 14101 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2023.

[39] Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. egg: Fast and extensible equality saturation. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021.

[40] Stephen Wolfram. The physicalization of metamathematics and its implications for the foundations of mathematics, Mar 2022.

[41] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy Liang, and Jure Leskovec. Deep bidirectional language-knowledge graph pretraining. 2022.

[42] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models for question answering. 2022.

[43] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. 2021.